

AD-A064 396

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO
DESIGN OF A SEL 86/LSI-11 INTERFACE MONITOR.(U)
DEC 78 J E BARALLI

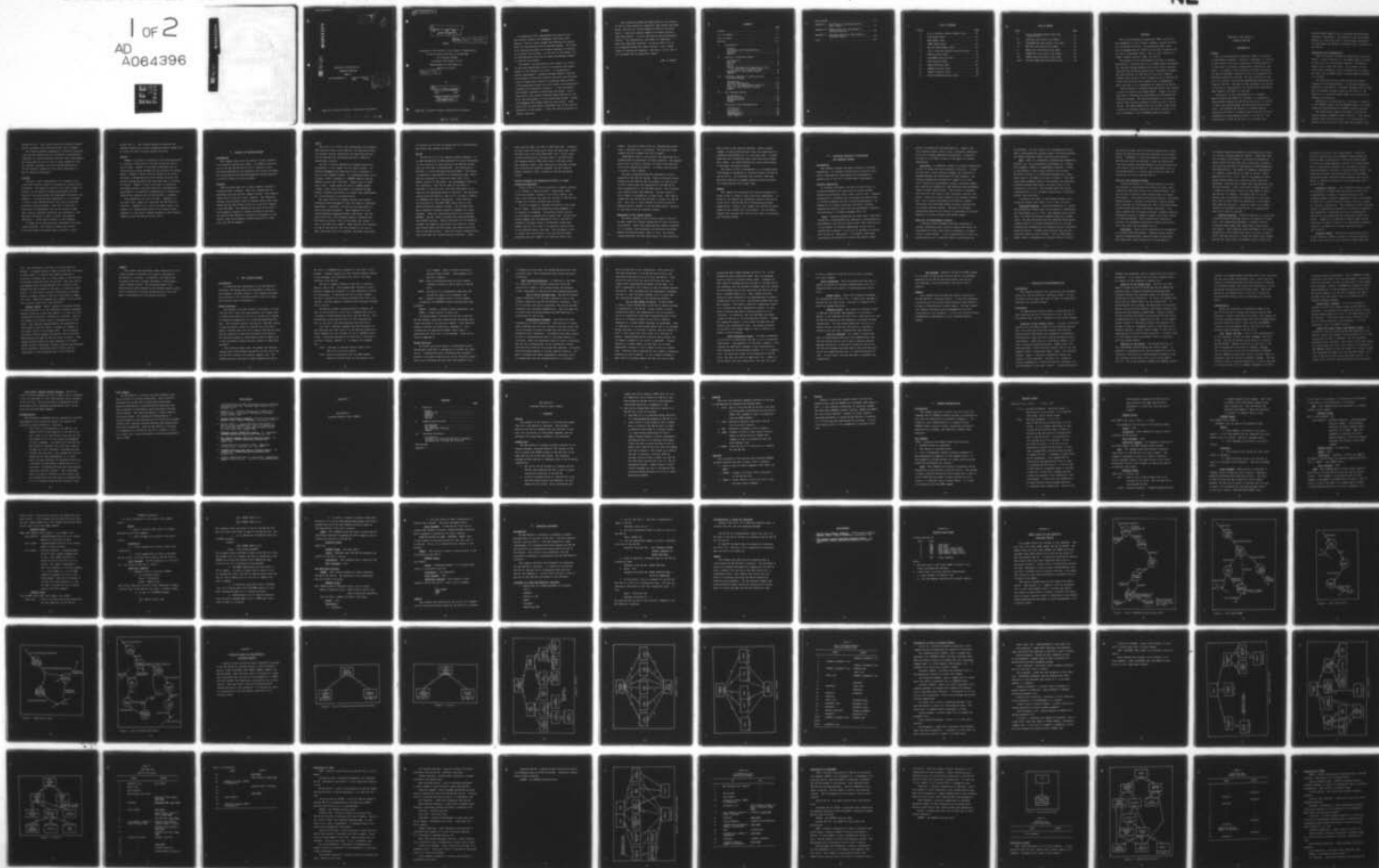
SCH--ETC F/6 9/2

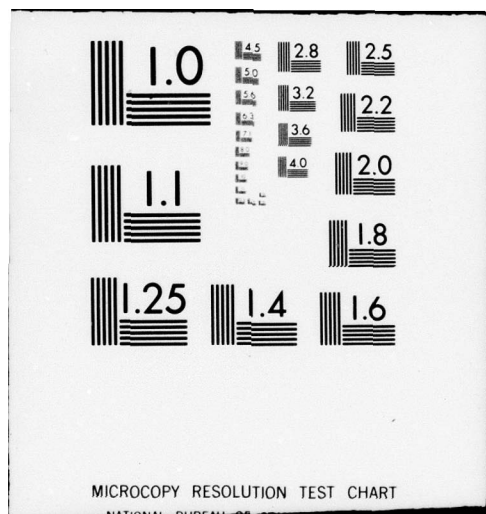
UNCLASSIFIED

AFIT/GCS/EE/78-9

NL

1 of 2
AD
A064396





LEVEL

①

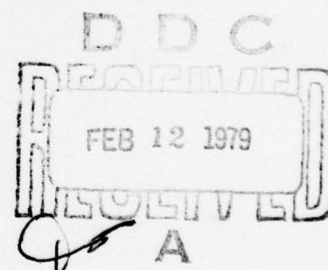
ADA064396

DDC FILE COPY

DESIGN OF A SEL 86/LSI-11
INTERFACE MONITOR
THESIS

AFIT/GCS/EE/78-9

Janet E. Baralli
Capt USAF



Approved for public release; distribution unlimited.

79 01 30 098

14

6 DESIGN OF A SEL 86/LSI-11
INTERFACE MONITOR.

9 Master's Thesis

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air Training Command
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

12 113 p.

by

10 Janet E. Barallig B.S.

Capt

USAF

Graduate Computer Systems

11 December 1978

ADDITIONAL IN	
ATIS	With Section <input checked="" type="checkbox"/>
UNIT	With Section <input type="checkbox"/>
BY	<input type="checkbox"/>
DATE	
REPRODUCTION AUTHORITY CODE	
1st	
2nd	
A	

Approved for public release; distribution unlimited

012 225

elf

Preface

My experience with "programming" has shown me that I have wasted valuable time debugging inefficient code because I have not spent enough time defining the requirements and considering a well-structured design. The formal tools becoming available to software engineers is helping to alleviate this problem. No one tool is the answer, but by blending these tools one can make considerable headway in tackling the problem.

The design I am presenting in this thesis is a result of devoting considerable time to the analysis and design phases of the software life cycle. In attempting to define requirements I stumbled through SofTech's SADT and Tom De Marco's Structured Analysis and found that drafting a user's manual best served as my analysis phase product. I had exceptionally good luck with Yourdon and Constantine's transform and transaction analyses. I also used Parna's and Jackson's methods in developing the design. Use of these tools, with sound structured design heuristics, led me to the development of a well-structured design. Coding and debugging this design should be much easier. Even errors will be more easily correctable due to the cohesive modularity of the design. I feel the resulting product is indeed a good one.

I must extend my thanks and appreciation to my sponsors at the Air Force Materials Laboratory, Mike Dennis and Frank Beitel, and also to Lt Dave Summer at AFML for his assistance. I must give special thanks to my thesis advisor, Capt Peter Miller. I am not sure how he survived being my advisor. But miraculously, the thesis is completed. I must also thank Maj Alan Ross. He put up with a lot of my struggling during this thesis project. And I cannot forget to thank Rusti Gaudreau. She had so little time to do a splendid job in typing this report.

Janet E. Baralli

Contents

	Page
Preface	ii
List of Figures	vi
List of Tables	vii
Abstract	viii
I. Introduction	1
Problem	1
Constraints and Considerations	2
Approach	3
Outline	4
II. Analysis of Existing System	5
Introduction	5
Overview	5
LSI-11	6
SEL 86	7
Current Procedure for Operating an LSI-11 in a Data Collecting Experiment	8
Weaknesses of the Current System	9
Summary	10
III. Techniques Employed in Developing the Interface Design	11
Introduction	11
Software Engineering	11
Tools for the Requirements Analysis	12
Tools for the Structural Design	14
Summary	18
IV. The Interface Design	19
Introduction	19
Design Structure	19
Design Decisions	21
Summary	26
V. Conclusions and Recommendations	27
Introduction	27
Conclusions	27
Implementation	28
Recommendations	30
Final Summary	31

Bibliography	32
Appendix A: SEL 86/LSI-11 Interface Monitor User's Manual	33
Appendix B: Bubble Charts for SEL 86/LSI-11 Interface Monitor	54
Appendix C: Structure Charts for SEL 86/LSI-11 Interface Monitor	60
Vita	101

List of Figures

Figure		Page
1	LSI-11 Interface Monitor Bubble Chart	55
2	LOAD Bubble Chart	56
3	SEND Bubble Chart	57
4	TRANS Bubble Chart	58
5	SEL 86 LSINTR Bubble Chart	59
6	LSI-11 Interface Monitor Structure Chart . .	61
7	LOAD Structure Chart	70
8	FILE/SEND Structure Chart	78
9	RUN Structure Chart	82
10	TRANS Structure Chart	83
11	IMCHAR Structure Chart	86
12	IMEXIT Structure Chart	88
13	SEL 86 LSINTR Structure Chart	89

List of Tables

Table		Page
I	LSI-11 Interface Monitor Data and Control Flow	66
II	LOAD Data and Control Flow Table	73
III	FILE/SEND Data and Control Flow Table	79
IV	RUN Data and Control Flow Table	82
V	TRANS Data and Control Flow Table	84
VI	IMCHAR Data and Control Flow Table	87
VII	IMEXIT Data and Control Flow Table	88
VIII	SEL 86 LSINTR Data and Control Flow	95

Abstract

The Air Force Materials Laboratory (AFML) uses LSI-11 microcomputers as one of several computer systems available for collecting test data. For conducting these tests, LSI-11 programs must be loaded into and data collected from the LSI-11 using paper tapes. Data is later stored on a larger computer system at AFML, the SEL 86.

The purpose of this investigation has been to design a SEL 86/LSI-11 interface that will automate manual procedures. The interface design enhances the current LSI-11 system by providing the following capabilities: load binary programs and data residing on a SEL 86 file into LSI-11 memory; transmit data stored in LSI-11 memory to one or more SEL 86 files; and place the LSI-11 memory into a transparency mode such that it is a peripheral as viewed by the SEL 86.

The principles of software engineering have been applied in both the analysis and design phases. Formal tools have been used in defining the requirements and developing the structured design. The resulting design is an interface monitor with software residing on both the LSI-11 and the SEL 86. The added capabilities are provided using either a series of commands entered at the LSI-11 console or as call statements in LSI-11 FORTRAN compiled programs.

DESIGN OF A SEL 86/LSI-11

INTERFACE MONITOR

I. Introduction

Problem

The Air Force Materials Laboratory (AFML) is involved in experimental research. Testing is conducted in a variety of areas within the field of materials including research in such areas as corrosion crack growth, electronics, and lasers. In support of this research, laboratory personnel use Digital Equipment Corporation LSI-11 microcomputers for some data acquisition testing. The data collected on the LSI-11 is stored on the Systems Engineering Laboratories 86 computer (SEL 86). The data is then analyzed on the SEL 86 or the data is transferred to the Control Data Corporation 6600 computer (CDC 6600) for analyses.

The purpose of this investigation is to design a general-purpose SEL 86/LSI-11 interface. At present all communication between the SEL 86 and any LSI-11 is limited to paper tape inputs/outputs and associated manual operations. The proposed SEL 86/LSI-11 interface enables laboratory personnel to transmit data to and from an LSI-11 automatically using keyboard inputs at the LSI-11. The modification of both SEL 86 and LSI-11 software also

provides added capabilities in designing and performing laboratory experiments. The purpose of the new software design is to meet the needs of the personnel performing the experiments by adding an automated LSI-11 interface to the SEL 86.

Constraints and Considerations

There are several limitations imposed upon the system design. A major constraint is the memory space available on the LSI-11 for the modified software. The maximum memory size is 8192 (8K) words. In an effort to stay within these boundaries, LSI-11 software must be useful, but short and straightforward in keeping memory overhead to a minimal.

Any modifications must remain within the present framework of existing hardware. No hardware changes are permitted on the SEL 86. Available equipment must be used. The only hardware change allowed on the LSI-11 is the addition of a read-only memory (ROM) to store the LSI-11 software modifications.

Modification to existing LSI-11 software is required in the FORTRAN library and the FORTRAN compiler in recognizing and interpreting new FORTRAN callable routines.

The communication support available is the SEL 86 Terminal Support Subsystem (TSS) (Ref6:2-1). This system imposes some restrictions since it is constrained to operate at the slow rate of 300 baud and transmits ASCII

characters only. This latter restriction requires encode/decode procedures into both SEL 86 and LSI-11 software.

The limited computer background of laboratory personnel using the new software must be considered in determining trade-offs in system complexity and user input requirements. The system must be easy to use for those unfamiliar with any computer interface, yet allow enough flexibility to provide powerful capabilities for those experienced in SEL 86 interface procedures.

Approach

The goal of this investigation is to design a well-structured interface system to automate the current LSI-11 procedures. The approach consists of two major stages: an analysis of the interface system requirements and the design of the system using the requirements obtained by the analysis. In analyzing the requirements it is necessary to have a clear understanding of the current operating procedures and environment. Once this is achieved, a definition of requirements for the new system is made. This involves continued interviews with personnel at the Materials Laboratory to gain a concise requirements definition. A draft user's manual is then prepared.

A design can be developed once requirements have been clearly defined. This design is based upon various structured design techniques used to produce a "good"

design (Ref 4). The finished design incorporates the expanded capabilities without degrading present capabilities by using the "best" possible design structure.

Outline

Chapter II presents an analysis of the existing system. It includes a description of the SEL 86 and the LSI-11s used in AFML. Chapter II describes techniques used in developing the design through both the analysis and design phases. Chapter IV deals with the design of the actual interface, considering the overall design structure and specific design decisions that have been made in creating the design. Chapter V sites conclusions in examining the design product, specifies particular implementation requirements, and presents recommendations to enhance the capabilities acquired with the SEL 86/LSI-11 interface.

Three appendices are also included. Appendix A is the draft user's manual for operating the SEL 86/LSI-11 interface. Appendix B is a series of bubble charts (see Chapter III) representing output of the analysis phase. Appendix C is the actual software designs of the interface needed for both the SEL 86 and the LSI-11.

II. Analysis of Existing System

Introduction

This chapter deals with the analysis of the system as it is presently operating at AFML. It presents a description of how the SEL 86 and the LSI-11 operate in data collecting experiments and summarizes the current procedure for collecting and storing data generated during execution of an LSI-11 data-collecting program.

Overview

AFML personnel make use of three computer systems in conducting their research. Many test computations are performed using the ASD CDC computer system. This computer system can be viewed as a host computer. The laboratory operates and maintains a SEL 86 computer which, when interfaced with the CDC system, serves as a satellite computer to the CDC 6600. The laboratory also owns eleven LSI-11 microcomputers located in several building on the base. By using the SEL 86/LSI-11 interface as presented in this thesis, these micricomputers become satellites to their host, the SEL 86 computer.

LSI-11

The LSI-11 is a 16-bit, byte addressable microcomputer. AFML operates eleven such computers: one dedicated solely for program testing, assembling, and compiling purposes; and ten employed for data-gathering while conducting experimental research.

The LSI-11 system, dedicated to program testing and development, is the largest of the eleven systems. All testing, debugging and compiling of LSI-11 programs is done on this computer system. Its operating system, RT-11, includes a PDP-11 assembler, a FORTRAN compiler and the FORTRAN library (Ref 3:681-693). Memory size is 32768 words (32K). Floppy disks are used to augment memory storage. Both inputs and outputs are handled through either a keyboard and printer or a paper tape reader/punch. All I/O processing is interrupt-driven.

The other ten LSI-11 computer systems are uniquely configured system designed to meet the needs of individual experiments. Currently, six LSI-11s are being used in data-collecting procedures. These computers all have limited memories ranging from 8K to 28K words. The computers have only a few standard features: keyboard with CRT scope or printer, fast or slow paper tape reader, a line to the SEL, and a modem. Other features are available to some of the LSI-11s, but not standard to all ten of them: real-time clock; X-Y recorder, and paper tape punch.

(at present all six LSI-11s being used for collecting data have paper tape punches available.)

SEL 86

The SEL 86 is a 32 bit, general purpose computer. It is used extensively by AFML personnel for local processing. It also serves as an input device to the ASD CDC system, like the 1700 used in building 640. Job processing is handled in a real-time operating environment. The system is essentially time-shared in an interactive mode. Batch jobs are processed on a priority basis (Ref 5:1-6). Memory consists of 96K words. Two 100-megabyte disks are also available. Over 75% of space on the disks is available to the user since less than one-fourth of the space for the operating system is reserved. The operating system, Real-Time Monitor, (Ref 8:1-1) includes a compiler for FORTRAN and a basic interpreter. Cross-compilers are available for the INTEL 8080 system and two less sophisticated SEL computer systems. The systems 85186 Macro Assembler processes assembly language into object programs. There are two pre-processors for structured FORTRAN. The SEL library includes the standard FORTRAN and CALCOMP routines. Devices available for I/O handling include: two magnetic tape drives, one each for 7-track and 9-track tapes; one fast paper tape reader and punch; and two 600 wpm printers. There are sixteen communications lines available for interfacing with terminals. These

lines operate under 110, 300, or 9600 baud rate. Presently thirteen lines are being used: three 110 baud lines, three 300 baud lines, and seven 9600 baud lines. Future plans include converting two 110 baud lines to 300 baud lines and changing several 9600 baud lines to 300 baud lines. Easy terminal access to SEL functions is available through use of the SEL 86 interface package, the SEL Terminal Support Subsystem (TSS), residing in the SEL operating system.

Current Procedure for Operating an LSI-11 in a Data Collecting Experiment

Using an LSI-11 in data collection to support research is, at present, a manual process. Generating a data-collecting program, loading it into LSI-11 memory, and later collecting the test data for transfer to SEL 86 file storage are a series of paper tape actions.

An LSI-11 program is designed and tested on the large LSI-11 system. Once the program has been debugged, it is compiled or assembled. The absolute binary version of the program is punched out on paper tape. Storing this binary program into the appropriate LSI-11 consists of reading in two paper tapes: a "bootstrap" loader that enables the LSI-11 to read in an absolute load followed by the absolute binary load tape. This procedure varies in length from five minutes to one and one half hours, depending upon the length of the absolute binary load

module. The entire effort fails if, sometime during the load, a checksum error is detected. The absolute binary program must be reloaded into LSI-11 memory.

Experimental data is collected by the executing LSI-11 program while an experiment is being conducted. The program may collect raw data or process a string of raw data and store only refined data points. In either case the data is stored in LSI-11 memory.

The data collected during the experiment is later analyzed on the ASD CDC system. Since an interface exists between the CDC 6600 and SEL 86 systems, all data stored in LSI-11 memory must be transferred to the SEL 86 for later transmission to the CDC 6600 system. This procedure also requires paper tape handling. The data stored in LSI-11 memory is punched out on paper tape. That tape is loaded into the SEL 86 and the data is stored on a SEL 86 file. This file is later checked for errors by SEL 86 software before transfer to the CDC 6600 system. Analysis is then done using CDC resident software.

Weaknesses of the Current System

The major weakness of the current system is the use of paper tapes for program loading and all data collection and transfer. Paper tape processing has inherent problems. It is a manual, time-consuming, and unreliable procedure. Reading and punching paper tapes is slow. The current loading procedure can take large amount of time especially

when errors in the load are detected. Simple program changes are major modifications since the program must be recompiled and punched out on a new paper tape. As paper tapes age due to handling, paper tape read errors become more common. All these problems are costly shortcomings of the paper tape-based system.

The intent of this investigation is to overcome these shortcomings by designing an interface between the SEL 86 and an LSI-11. The interface will provide a much more efficient handling of program loading and data transfers by eliminating the use of paper tape.

Summary

This chapter has described the current procedure for using an LSI-11 system in data collection experiments. In doing so, the chapter has presented a general description of the LSI-11 and the SEL 86 in determining interface capabilities. Finally, this chapter has covered the weaknesses of the current LSI-11 procedures. The following chapter will describe the various tools used in developing the interface design.

III. Techniques Employed in Developing the Interface Design

Introduction

This chapter presents the basic concepts of software engineering. It includes a description of specific tools used in analyzing the requirements and designing the capabilities of the LSI-11 system under investigation.

Software Engineering

As software continues to be more and more costly in system developments the application of sound software engineering principles plays a more critical role. Software engineering is concerned with the design and construction of software programs and their related documentation. By making use of various disciplines available, software engineering concerns itself with the development, operation, and maintenance of software packages (Ref 2:3).

Goals. Software engineering has seven goals: efficiency, reliability, understandability, generality, maintainability, modifiability, and utility (ease of use) (Ref 4 and 9:9-12). In any product of software engineering, as well as all engineering in general, it will not be possible to successfully attain all these goals. For example, efficiency constraints and reliability criteria may greatly impair

efforts for generality and modifiability. However, the test software product results from a successful combination of several software engineering disciplines. Trade-offs are made in an attempt to meet all the goals of software engineering.

Life-cycle. An important concept of software engineering is that it covers the entire software life-cycle. It does not end with a tested software package, but continues on through redesign and modification of that software. Any software package goes through five stages in its life-cycle: analysis of requirements; design; coding and debugging; testing and integration; and operations and maintenance (which includes modification). This thesis is concerned with the first two states in the life-cycle, analysis and design. These are critical phases in any software development. Well defined requirements followed by a highly structured, well-developed design are major contributions to a useful software product. The goals of software engineering cannot be met without a structured, disciplined approach in software analysis and design stages.

Tools for the Requirements Analysis

A software system designed using the techniques of software engineering must carefully examine and define the requirements of that system before attempting to actually create a system design. A clear representation of what the specifications are is essential before a good design can

be developed. For this thesis, two techniques have been used in performing this analysis phase: Structured Analysis and Design Techniques (SADT) and Structured Analysis.

SADT. SADT is a comprehensive methodology designed by SofTech, Inc. for analyzing the requirements of a system. The language of SADT is a diagramming technique. A model of the problem is built using a precise set of rules. The model consists of a structured decomposition, using these rules to introduce new levels of detail. This technique shows component parts of the problem, the interrelationships between these parts, and their place in the hierarchical structure. The finished model has several characteristics: it is top-down, hierarchical, structured, modular, and functional.

It defines the "what" of the problem without introducing the design "how" (Ref 1:1-1-2-1 and 7:2-1-2-3).

Structured Analysis. Structured Analysis is a technique described by DeMarco that can be applied in specifying the requirements of a problem. This approach uses several tools: the Data Flow Diagram (commonly known as a bubble chart), the Data Dictionary and the Transform Descriptions. A bubble chart is a network representation of a system, showing major decomposition of functions and all interfaces among the pieces. A bubble chart portrays data and the processes which act upon the data. Like an SADT model, a bubble chart is decomposed into several levels of detail.

The Data Dictionary and the Transform Descriptions clearly define the data and the processes presented in the bubble charts. Techniques of structured English, decision tables, and decision trees are employed to make the definitions clear and concise, avoiding ambiguities and inconsistencies often found in written English. By using these tools of structured analysis, a concise specification of what the system must do can be achieved (Ref 4).

Tools for the Structured Design

Structured design is a technique employed to assist the designer in determining the modules and their interconnections which best solve a well-stated problem. Once analysis has successfully been accomplished, the design phase begins. Like SADT and Structured Analysis, Structured Design produces a top-down, hierarchical, modular design. Tools available for developing the "best" structured design include design heuristics and design techniques. This thesis employs the following techniques: transform analysis, transaction analysis, Jackson's method, and Parnas' method of structured design. Both transform and transaction analyses are techniques developed by Yourdon and Constantine; the latter two techniques are not.

Heuristics. There are five guidelines to be employed in any good structured design. Cohesion between modules should be maximized while coupling should be minimized. This means that a good design has a strong degree of

relatedness among the elements of each module while there are minimal coupling among the modules. Trade-offs often must be made in the design in following this guideline. A module should be sized such that it processes a complete function, but maintains a high degree of understandability and modifiability. The number of immediate subordinates to a module, called fan-out or span of control, should generally be from 1 to 10. Fan-in, the number of subordinate modules which call a specific module should be maximized. Multiple fan-in means that some duplicate code has been avoided. Lastly, the scope of effect should be a subset of the scope of control of the module in which the decision is located. This means that all of the modules that are affected, or influenced, by a decision. The scope of effect should be subordinate to the module that makes the decision. (Scope of control of a module consists of the module itself and all its subordinates (Ref 4 and 9:76-125, 148,169)).

Transform Analysis. Transform analysis is a particular structured design technique based on the analysis of data flow. The system being designed is viewed as central transforms which digest and create major system inputs and outputs. This technique takes advantage of the overall perspective of the problem and leads to a fully, or almost fully, factored structure in which the lowest level modules perform the "work" while intermediate levels control and

coordinate the work of their subordinates. Transform analysis uses the data flow diagrams created during the analysis phase in generating a "first-cut" design model. A "first-cut" design generally consists of an input module and its hierarchical decomposition, followed by central transform modules, and lastly a factored output module. By using the previously described heuristics of structured and making design trade-offs where necessary, the initial design is refined to develop a final design model (Ref 4 and 9:171-185).

Transaction Analysis. Like transform analysis, transaction analysis is based upon the analysis of data flow. A supplementary strategy to transform analysis, this technique handles processes suggested by data flow diagrams in which one of several possible outputs occur as a result of a process (transaction). The transaction center functions consist of: getting the transaction in raw form, analyzing the transaction to determine its type, dispatching the transaction, and completing the processing of each transaction. Common use of transaction centers might include conversion of input or output to its appropriate formatted version or validation of an input transaction (Ref 4 and 9:202-221).

Jackson's Method. Unlike both transform and transaction analyses, Jackson's method of structured design is based on the analysis of data structure rather than data

flow. The structures of the data to be processed are defined. A program structure based on these data structures is then formed. In essence this approach develops a hierarchy of modules that is a minor image of the hierarchy of the data associated with the problem. When this one-to-one mapping cannot be made between the structure of the program and the structure of the data, a "structure clash" exists. If this occurs, multiple hierarchies of modules, known as program inversion, must be established to handle the clash. Additional programs are created to work around the structure clash (Ref 4 and 9:223-227).

Parnas' Method. Parnas' method of structured design is often referred to as the "information hiding" technique. Parnas' principle is that each module is to have as little information as possible to define its interface. Details are hidden to the lowest levels of decomposition of the model. This improves maintainability and modifiability since details that are likely to change are "hidden" at the lowest levels of the design model. This eliminates the need to filter changes through the entire hierarchy. Modifications are made at the low levels of detail. Such design decisions which are good candidates for Parnas' method include: formatting, linking, storing, and modifying data structures; formatting control blocks; and the sequencing of item processing (Ref 4 and 9:228-230).

Summary

This chapter has presented a short description of the formal analysis and design tools used in creating the SEL 86/LSI-11 interface. In addition, the chapter has introduced the basic concepts and goals in developing a "good" software system. The following chapter will describe the overall design of the interface monitor along with a discussion of various design decisions that were made in developing the final design structure.

IV. The Interface Design

Introduction

In analyzing the requirements of the SEL 86/LSI-11 interface, the design phase has determined a need for a multicomputer interface monitor. This chapter describes the structure of the interface monitor, how it is used, and decisions made in designing it.

Design Structure

The purpose of the SEL 86/LSI-11 interface is to provide the LSI-11 user with some powerful, yet simple tools to improve utilization of the LSI-11 as part of a network of computers designed for collecting and analyzing test data. The interface enables the user to: load data stored in absolute binary format on a SEL 86 file into LSI-11 memory; transmit data from LSI-11 memory onto SEL 86 files; and use interactive functions available on the SEL 86. The interface design is constrained by LSI-11 memory size. It must be general purpose and must require no additional equipment.

The resulting design uses: the modems and 300-baud communication lines between the SEL 86 and the LSI-11 and the SEL 86's software for terminal support, TSS. The capabilities are realized by either keyboard entries at

the LSI-11 or FORTRAN CALL statements in the user's LSI-11 program. Software support for the interface design consists of two programs: one residing on the LSI-11, the other residing on the SEL 86.

The LSI-11 support software is the LSI-11 Interface Monitor (LSI-11 IM). This program must interpret a user's keyboard command and dispatch it to the appropriate routine for processing. In executing the user's input command, the LSI-11 controls all interaction between the LSI-11 and the SEL. The SEL program is executed and terminated through LSI-11 control.

The SEL 86 resident interface monitor software, LSI NTR, is initialized for those LSI-11 commands that call for a transfer of data between the LSI-11 and the SEL 86. This program must access SEL 86 files for both transmitting data to the LSI-11 and receiving data from the LSI-11.

Five LSI-11 keyboard commands have been designed for providing the user with control of the desired capabilities. These commands are also callable by LSI-11 FORTRAN conventions. A detailed description of their use is provided in the User's Manual, Appendix A. In summary the commands are:

LOAD - load data in absolute binary format from a
SEL file into LSI-11 memory;

FILE - used in conjunction with the SEND command.

Specifies SEL 86 files for storing data from

LSI-11 memory. Data is stored in binary or absolute binary format. (See Appendix A of the User's Manual.)

SEND - used in conjunction with the FILE command.

Transmits a block of LSI-11 data to a SEL 86 file;

TRANS - put LSI-11 in a transparency mode such that the SEL 86 views it as a terminal; and

RUN - execute a program stored in LSI-11 memory.

In addition two FORTRAN callable routines have been designed:

IMCHAR - transmit a string of ASCII characters; and

IMEXIT - return control to LSI-11 IM.

Two design models, one each for the LSI-11 and the SEL 86 software, have been created using the design techniques described in the previous chapter. Their detailed design structures are specified in Appendix C. Also included are the appropriate bubble charts used in generating the "first cut" of the designs. These can be found in Appendix B.

Design Decisions

No design can be successfully accomplished without making many trade-offs in attempting to produce the "best" design. A dominating factor influencing the interface design is the need to keep the use of the interface simple while still providing the user with some powerful tools.

8 In keeping with this goal, the design decisions fall into two broad areas: data transmission and receipt and error processing.

Data Transmission/Receipt. In addition to the need for simplicity, LSI-11 memory constraints force the selection of simpler, less sophisticated, and normally slower, approaches in dealing with data handling decisions.

Use of TSS at 300-baud rate. The SEL 86 Terminal Support Subsystem (TSS) has been selected to use as the terminal system driving the SEL 86/LSI-11 interface. Since it is available and easily accessible, this choice greatly simplifies the interface communication. The slow baud rate is forced upon the design because the 9600 baud rate is unavailable for LSI-11 use.

Encode/decode procedure. TSS limits all data transmission to ASCII characters. This does not in itself cause problems since the bits of data to be sent across TSS can be released in groups of eight bits and thus be interpreted as ASCII characters. A problem arises when a group of bits is interpreted by TSS as a carriage return or a line feed. These two characters serve as control characters and are not considered part of the transmission itself, thus data bits would be lost using this approach. An encode/decode procedure appears to be the simplest solution. All data is encoded into ASCII alphanumeric characters prior to transmission and then decoded before it is stored at

the receiving end of the transmission. This occurs for both data being sent to the SEL 86 from an LSI-11 and data being received by an LSI-11 from the SEL 86. Since the LSI-11 has 16 bits per word and 8 bits per byte, a simple 4-bit encode/decode procedure can be used. All ASCII transmissions will consist of one of sixteen possible characters. This procedure has the advantage of enabling the user to "tap" the data transfer line to verify that the transmission appears to be working successfully.

Use of data buffer on LSI-11. No data buffer is used to store encoded data on the LSI-11 before the data is transmitted to the SEL 86. Using a buffer to hold encoded data before the communication line is available can be of significant value when transmitting data to the SEL as it is being collected on the LSI-11; however, drawbacks of a single buffer outweigh its advantages. Data is expected to be collected much faster than it is released. Unless the data is collected in bursts, once the buffer is full it will remain full until there is no new data to buffer. Double-buffering can alleviate this problem, but its memory overhead is too costly to implement. Another problem of a single buffer is that there is no fixed addressable location in LSI-11 memory to use as a buffer area. Absolute binary programs are not loaded in sequential addresses of LSI-11 memory. It will require overhead to keep track of available memory and this can only be done

program has been loaded through the LSI-11 IM. If the program has been loaded from paper tape, no automated record is kept of available memory space. Overhead for both types of bookkeeping could be kept at the SEL end of the interface, but the problem of buffer size still exists. No fixed buffer size can be used since the LSI-11s do not have a standard memory size. Thus, buffer size must become an input parameter or an algorithm must be used to automatically set its size. The problem becomes more and more complex. This buffer, would be a significant factor if it affected all data transmission; however, it only poses a problem when data is being released as it is collected. In actuality, the slow 300-baud rate makes release of data as it is collected most unlikely. Though the capability exists, it will rarely be used with the present slow transmission lines. The design structure of the LSI-11 IM, though, makes it easy to modify if a buffer is later desired.

Data transmission format. All data transmitted by the LSI-11 or received by the LSI-11 is in an absolute load format. (See Appendix A of the User's Manual.) This requires an additional overhead in data processing for both sending and receiving data; however, it is a powerful tool. By using this format for storing data on SEL 86 files, the user can store an image copy (i.e. a dump) of LSI-11 memory and, more importantly, the user can store

an LSI-11 program on a SEL 86 file for later reloading into LSI-11 memory.

Error Processing. Error processing has been kept as simple as possible to minimize overhead while still providing the user with control of processing after errors occur.

Format errors. Input LSI-11 keyboard commands are validated by the LSI-11 IM. A simple error message is printed for any input error. The user is required to re-enter the command correctly.

Checksum errors. The trailer of a block of data in absolute load format (See Appendix A of the User's Manual) is the checksum for that block. Both the LSI-11 and the SEL 86 programs use the checksum to validate the data transmission. For data being loaded into the LSI-11 in several blocks, the user has the option to continue or terminate the load once a checksum has been detected.

Time out warnings. As a check to ensure that data is being transmitted across the communication line, both the SEL 86 and the LSI-11 have a time-out warning message printed out to the user's console at the LSI-11. The time out message specifies that no data has been sent across the communications line in a specified period of time. At this point, the user may want to terminate the transmission.

SEL messages. Whenever the SEL 86 LSINTR program is in a mode to receive data from an LSI-11, any messages generated by the SEL 86 operating system, such as file error messages, will be printed at the user's LSI-11 console.

Summary

This chapter has described the overall structure of the SEL 86/LSI-11 interface monitor. It has also presented a discussion of the various design decisions that needed to be made in developing the current design. The next chapter will present conclusions and recommendations from the investigation of an interface. It will also discuss several implementation requirements for the interface monitor design.

V. Conclusions and Recommendations

Introduction

This chapter presents final conclusions and recommendations for the SEL 86/LSI-11 interface system. It also discusses various steps that must be taken to successfully implement the interface design.

Conclusions

In examining the design product, conclusions can be made regarding the use of the analysis tools, the use of the design tools, and the usability of the current design.

Usability of the Analysis Tools. In using a structured approach for the analysis phase of the interface development, both SADT and Structured Analysis were applied. Though several SADT models were attempted, none seemed to satisfactorily state the exact problem and define actual requirements. SADT could not successfully handle a top-level transaction-centered process. Bubble charts were then generated. These were more helpful, but only at the highest levels in separating input, output, and transform modules. Again, a top-level transaction limited the success of this technique. The best tool, though not formally regarded as a structured analysis technique, was the development of the user's manual. In defining specific

commands and parameters, precise capabilities were clearly documented. In its final form, the user's manual covers all aspects of the interface's capabilities.

Usability of the Design Tools. Once the top level modules of both the LSI-11 IM and the SEL 86 LSINTR were modeled using a transaction-centered design approach, transform analysis was extremely useful in designing the processing of each transaction. All first cut designs were based upon the bubble charts for the transactions. In developing the final designs, the heuristics of structured design, along with both Jackson's and Parnas' methods, were applied and successfully accomplished. Jackson's method of data structure was useful in processing the data blocks transmitted across the communication lines. In all models Parnas' method of "information hiding" was applied in an attempt to keep all changeable detail at the lowest level, wherever possible. Parnas' theory seemed to "fall out" in applying good structured design principles. In short, each of these design tools was extremely useful in building the final design models.

Usability of the Design. The design satisfies all the requirements defined during the analysis phase. In addition to implementing the desired capabilities into an interface, the design has remained general purpose so that the interface can be used on any of the eleven AFML's LSI-11s. Use of the keyboard commands and FORTRAN callable

routines is straightforward and simple while still providing the user with useful and powerful tools. Since the principles of structured design have been applied in creating the interface design, the system, once operational, can be maintained and modified with relative ease. Additions, deletions, and modifications to the LSI-11 IM and the SEL 86 LSINTR software can be accomplished with minimal effort.

Implementation

Before the interface is permanently added to the SEL 86 and the LSI-11 systems and actively used by AFML personnel, several steps need to be taken: code, debug, and test the interface design; modify the user's programs where desired; modify the LSI-11 linker utility and FORTRAN library on the development LSI-11; and store the user's absolute binary programs on SEL 86 files.

Code, Debug, and Test. Prior to full implementation of the interface, it must be coded, debugged, and fully tested. The SEL 86 routine can be written in standard SEL 86 FORTRAN. Dynamic allocation of new files in the SEL 86 program must be written in assembly language. Most of the LSI-11 IM can also be developed in FORTRAN. Several command routines which allow a variable number of parameters must be written in assembly language. Once the system is found to work satisfactorily, it may be desirable to recode the LSI-11 IM in assembly language in an effort

to decrease memory requirements. Use of FORTRAN is highly preferred at the AFML, since most programmers, particularly contractor personnel, do most of their work in that higher order language. The use of FORTRAN will greatly increase understandability of the documentation, particularly for the SEL 86 program. For the LSI-11 program, the PDP-11 assembler has some useful tools for documenting LSI-11 assembly language programs.

Modify Users' Programs. All LSI-11 programs as currently written will execute using the SEL 86/LSI-11 interface. For programs that require the release of data during program execution, a call to the SEND routine will be required. If the user wants control returned to the LSI-11 IM upon completion of the program, it will be necessary to call the IMEXIT routine.

Modify LSI Linker Utility and FORTRAN Library. In linking the various routines during the generation of an absolute binary load program, the development LSI-11 linker utility must be able to recognize any call statements designed by this interface. The linker utility must recognize that either the call to the routine is an external in the FORTRAN library or that the linker is to access a table that specifies the address of the particular memory location in the LSI-11 that stores the called routine. In either case, the RT-11 operating system must be modified.

Store Users' Absolute Binary Programs. Before any loads of binary programs into LSI-11 memory can be attempted, it will be necessary to store some absolute binary programs on SEL 86 files. This can be easily accomplished on the development LSI-11 system by accessing the LSI-11 IM and using the FILE and SEND commands.

Recommendations

The following recommendations are presented as possible improvements that can greatly enhance the capabilities available using the SEL/LSI-11 interface.

1. Add a PDP-11 cross-compiler on the SEL 86. By adding a PDP-11 cross-compiler on the SEL 86 system, a user is not restricted to use only the one development LSI-11 for the coding and debugging of LSI-11 programs. Any AFML LSI-11 that has the SEL 86/LSI-11 interface can access this compiler through use of the interface's transparency mode.
2. Increase the baud rate. Even though the interface design has not been implemented, it is fairly easy to foresee problems in transmitting data over a 300 baud line when data is being sent as it is collected. Increasing the baud rate would greatly facilitate the data transfer. This improvement would also make it feasible to add a buffer area for encoded data since the option of sending data as it is collected would be more regularly used.

Final Summary

The SEL 86/LSI-11 interface has been designed using the principles of software engineering. After carefully analyzing the current LSI-11 system and establishing baseline requirements, structured design techniques were easily employed in developing a good structured interface monitor design. The resulting design is a good, modular, structured design. The coding and testing phases of the development of the interface will be relatively straightforward since carefully executed analysis and design phases have been accomplished. Once the SEL 86/LSI-11 interface is integrated into the SEL 86 and LSI-11 systems, it will be a useful tool in more efficiently collecting experimental data at AFML.

Bibliography

1. "An Introduction to SADT Structured Analysis and Design Technique," 9022-78R. Waltham, Mass: SofTech, Inc., November 1976.
2. Boehm, B. W. "Software Engineering," TRW-55-76-08. Redondo Beach, California: TRW Defense and Space Group, October 1976.
3. Digital Microcomputer Handbook (1977-78, Second Edition) Maynard, Mass.: Digital Equipment Corporation, 1976.
4. Miller, Peter E. Lecture Materials distributed in EE6.93 Software Engineering. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, Summer 1978.
5. Reference Manual SYSTEMS 86 Computer. Ft. Lauderdale: SYSTEMS Engineering Laboratories, February 1976.
6. SEL Terminal Support Subsystem Reference Manual. Ft. Lauderdale: SYSTEMS Engineering Laboratories, August 1977.
7. "Structured Analysis Reader Guide," 9022-73.2. Waltham, Mass.: SofTech, Inc., May 1975.
8. SYSTEMS 85186 Real-Time Monitor Reference Manual. Ft. Lauderdale: SYSTEMS Engineering Laboratories, October 1974.
9. Yourdon, Edward and Larry L. Constantine. Structured Design (Second edition). New York: Yourdon Press, 1978.

Appendix A

SEL 86/LSI-11
Interface Monitor User's Manual

Contents

	Page
I. Overview	
Purpose	
Capabilities	
Commands	
Routines	
Outline	
II. Command Specifications	
Introduction	
The Commands	
The Additional Routines	
Summary	
III. Operating Procedures	
Introduction	
Procedure for Using SEL 86/LSI-11 Interface.	
Considerations in Using the Interface	
Bibliography	
Appendix A	

SEL 86/LSI-11
Interface Monitor User's Manual

I. Overview

Purpose

The purpose of this manual is to provide user guidelines for a SEL 86/LSI-11 interface. This document specifies the LSI-11 commands that are available to the user, the constraints in using these commands, and the procedure for using these commands in the interface.

Capabilities

The SEL 86/LSI-11 interface software consists of two program packages: Interface Monitor (IM) resides on the LSI-11 system; and LSINTR resides on the SEL file of the same name for use by the SEL system. The interface package provides options to implement each of the following capabilities:

1. the LSI-11 can be treated as a terminal by the SEL 86, thus enabling the LSI-11 user to access interactive functions on the SEL 86;
2. an LSI-11 program stored on a SEL 86 file in an absolute binary format (see Appendix A of this manual for the format) can be transmitted and

loaded into LSI-11 memory; (NOTE: Data can also be loaded into LSI-11 memory as long as it has been stored on the SEL 86 file in the absolute load format specified in Appendix A.) and

3. data can be transmitted from LSI-11 memory to a SEL 86 file in one of two ways:
 - a. send data as it is acquired during execution of a data-generating program on the LSI-11; or
 - b. send a block of data stored in LSI-11 memory.Data is stored on the SEL 86 file in either an absolute load format or a binary format, i.e. exact binary duplication of LSI-11 memory without header or trailer information. Storing the data in an absolute load format allows the user to store any data in LSI-11 memory onto a SEL 86 file for later re-loading into LSI-11 memory. This format also enables the user to perform a selective dump by storing a block of LSI-11 memory on a SEL 86 file and later routine that file to a SEL 86 peripheral printer. Binary format is specifically designed for use in storing the data collected during execution of an LSI-11 data-generating program.

Commands

There are five keyboard commands available to the user in implementing the capabilities defined above:

1. FILE - open or close the SEL 86 file(s) to be used in storing data transmitted from the LSI-11; (NOTE: This command is used in conjunction with the SEND command.)
2. LOAD - perform an absolute load from a SEL 86 file into LSI-11 memory;
3. RUN - executive a program in LSI-11 memory;
4. SEND - transmit a block of data stored in LSI-11 memory to SEL 86 file(s); (NOTE; This command is used in conjunction with the FILE command.) and
5. TRANS - allow the LSI-11 to function as a terminal for the SEL 86.

Routines

Also available to the user are the following FORTRAN callable routines that may be used in LSI-11 programs:

1. three of the five above commands: FILE, LOAD, and SEND;
2. IMCHAR - transmit a string of ASCii characters to the SEL 86; and
3. IMEXIT - return control to IM on the LSI-11 upon exiting a user's program.

Outline

Chapter II specifies command formats for both the keyboard input and the FORTRAN call statement where applicable. It also explains constraints in using the command. The additional FORTRAN callable routines, IMCHAR and IMEXIT are also clearly defined. Chapter III covers specific operating procedures in using the SEL 86/LSI-11 interface, i.e. initializing and terminating the interface. It also lists several points to be remembered in using the interface.

II. Command Specifications

Introduction

This chapter explains in detail the use of the five commands and two FORTRAN callable routines in providing the user with the capabilities as defined in Chapter I. Included for each command/routine is: keyboard format, FORTRAN format, expected output, termination procedures, expected error messages, and additional comments when necessary.

The Commands

(NOTE: Interpreting command format specifications:

1. Imbedded blanks are permitted.
2. <CR> denotes a carriage return.
3. A set of parentheses denotes optional parameter(s).

The parentheses are NOT part of the command format. If the parameter is being omitted, all format specifications within the parentheses are omitted.)

FILE. This command initializes or terminates SEL 86 interaction in storing data on SEL 86 file(s). The command is used to specify the SEL 86 file(s) where LSI-11 data is to be stored and the format in which the data is to be stored, i.e. absolute load or binary format. It is used in conjunction with the SEND command.

Keyboard format.

FILE(^L_B, file₁, file₂, ... , file_n) <CR>

L or B - optional parameter. Specifies format in which data is to be stored: L for absolute load format, B for binary format.

file_n - optional parameter. $0 \leq n \leq 8$.

Name of SEL 86 file(s) where data is to be stored. A file command specifying at least one SEL 86 file must be input in order to initialize SEL 86 actions in storing LSI-11 data on SEL 86 file(s). An existing file will automatically be allocated. If the file does not exist, a new file will be dynamically created and allocated. In this case, standard SEL conventions must be used in naming the file. A maximum of eight files may be designated. This command must be used without designating any files in order to terminate SEL 86 actions. It can be used in this form as a keyboard input to override a FILE call in the user's program and thus disable any automatic data transmission from the program. In this form, the command can be input anytime during program execution to terminate data transmission. Entry of the

FILE keyboard command with different file specifications will NOT override those designated in a FILE call from the user's program.

FORTTRAN format.

CALL IMFILE (^L_B, file₁, file₂, ..., file_N)

The parameters are the same as the keyboard format.

Output. none.

Termination. FILE command/call must be used to terminate SEL 86 actions.

Error messages. none.

Additional comments. For examples of the use of the FILE command, see SEND command specifications.

LOAD. This command loads data stored in absolute load format on a SEL 86 file into LSI-11 memory. The command is designed to serve as a loader of absolute binary programs; however, any type of data can be transmitted and stored in LSI-11 memory as long as the data is in absolute load format.

Keyboard format.

LOAD, file (, offset) <CR>

file - name of file on SEL 86 where data to be transmitted is stored. The file must be an existing SEL 86 file.

offset - optional parameter. Signed/unsigned positive

or signed negative octal integer. This value is added to the load address passed in the data transmission. This sum becomes the absolute load address. When this parameter is omitted, its default value is zero.

FORTTRAN format.

CALL IMLOAD (file, offset)

Parameters are the same as the keyboard format.

Output.

1. Start address is printed at the user's console. If the load address was not completed, the start address value is invalid. (See Error messages below.)

2. Error messages are printed at the user's console.

Terminator.

1. If no errors occur during the load, termination is automatic.

2. When load error occurs, user decides if load is to continue or be terminated. (See Error messages below.)

Error messages. When an error is detected in loading a block of data into LSI-11 memory, a message is printed specifying which block has an error. The block of data in error has been loaded into LSI-11 memory; however, the user has an option to continue with the load or abort it by answering the following message printed at the user's console: CONTINUE LOAD? ENTER Y OR N.

If the load is to continue, Y is entered via the keyboard.
If the load is to be terminated, N is entered.

RUN. This command causes the execution of a program stored in LSI-11 memory.

Keyboard format.

RUN (, start address) <CR>

start address - optional parameter. Signed/unsigned positive integer. This parameter specifies the memory address where execution is to begin. When this parameter is omitted, the start address defaults to the start address obtained from the most recent LOAD command.

FORTRAN format. not applicable.

Output. none.

Termination. automatic. Control will NOT be returned to the IM unless the executed program ends with a call to IMEXIT. (See IMEXIT below.)

Error messages. none.

SEND. This command sends a block of data stored in the LSI-11 to the SEL 86 and stores the data on SEL 86 file(s). This command, in conjunction with the FILE command, is designed to transmit and store data on SEL 86 file(s) in one of two ways: as a single datum is being collected during execution of an LSI-11 program or as a

block of data. (For specific details, see Additional comments below.) A FILE command initializing SEL actions must precede a SEND command and a FILE command terminating SEL 86 actions must follow that SEND command.

Keyboard format.

SEND, data address, data length (, file number) <CR>

data address - unsigned/signed positive octal integer.
address of first byte of data.

data length - unsigned/signed positive octal integer.
number of bytes of data.

file number - optional parameter. Unsigned/signed positive octal integer ranging from 1 to 10_8 . This number specifies on which SEL 86 file the data is to be stored. The number corresponds to that file parameter of the FILE command, i.e. a file number of 3 in the SEND command corresponds to file₃ of the FILE command. When the file number is omitted, the value defaults to 1 and all data is stored on file₁ as designated by the FILE command.

FORTRAN format.

CALL IMSEND (data name, data length, file number)

data name - name of the data item being transmitted.

The data name must follow SEL 86

FORTTRAN conventions.

All other parameters are the same as the command format.

Output.

1. Data is stored on SEL file(s) in format specified by the FILE command.
2. Error messages are printed at the user's console.

Termination.

1. A FILE command may be used to abort data transmission.
2. A FILE command must be used to terminate SEL 86 actions in storing LSI-11 data on SEL 86 file(s).

Error messages. SEL 86 messages (i.e. file full, etc.) are output to the user's console.

Additional comments.

1. Examples on how to use this command
 - a. As a keyboard input:
Enter: FILE,B,BILL
Enter: SEND,723,100

This stores 100_g bytes of LSI-11 data, beginning at memory location 723_g, on the SEL 86 file, BILL, in binary format.

- b. As part of a FORTRAN program:

```
⋮  
CALL IMFILE (B,BILL,JAN)  
⋮
```

```

CALL IMSEND (BUF(1),2,1)
:
CALL IMSEND (BUF(3),4,2)
:

```

This program stores two bytes of data on the SEL 86 file, BILL and stores four bytes of data on the SEL 86 file, JAN.

c. As a combination of keyboard input and a FORTRAN program:

```

Program:  :
          :
          CALL IMSEND (BUF(1),6,1)

```

Enter: FILE,L,BILL,JAN,PETE

This program stores six bytes of data on the SEL file, BILL. The program will not transmit the data as specified on the IMSEND call unless the keyboard FILE command has been entered prior to the execution of the program.

2. The SEND command does not store data in LSI-11 memory. In order to ensure that no data is lost if transmission fails, the user's program should store the data in LSI-11 memory prior to any call to SEND in the user's program.

3. It should be noted that since a 300-baud rate line is being used, this FILE/SEND routine is slow when transmitting data as it is being collected.

4. Sending data as it is acquired requires that the user's program make a call to SEND each time a piece of data is collected.

5. In order to transmit a block of data after execution of an LSI-11 data-generating program, that LSI-11 program must print the data address and data length of that data block to the user's console.

TRANS. This command allows the user to access SEL 86 interactive functions by making the LSI-11 appear to be an interactive terminal to the SEL 86.

Keyboard format.

TRANS <CR>

FORTTRAN format. not applicable.

Output. Keyboard input and SEL 86 responses are printed at the user's console.

Termination. User keyboard entry: Control/D <CR>

Error messages. none.

The Additional Routines

IMCHAR. This routine transmits an ASCII character string to the SEL 86. The characters to be transmitted must be stored in an array.

FORTTRAN format.

CALL IMCHAR (name of character array, size of array)

Name of character array - name of first word of the
array storing the characters.

Size of array - number of words in the array.

Output. none.

Termination.

1. automatic.

2. User has option to abort transmission if time-out has occurred. (See Error messages below.)

Error messages. If waiting for clear line to release data exceeds 3 minutes, warning message requiring user's response is printed at the user's console:

TIME-OUT WAITING TO SEND. CONTINUE? ENTER Y OR N
If user wishes to continue waiting for the clear line, Y is entered. If not, N is entered and the routine is terminated.

IMEXIT. This routine is used to return control to the IM residing on the LSI-11.

FORTRAN format.

CALL IMEXIT

Output. Prompting message "--" is printed when control is returned to the IM.

Termination. not applicable.

Error message. none.

Additional comments. This routine is most commonly used at the end of a user's LSI-11 program:

```

:
CALL IMEXIT
STOP
END
```

Summary

This chapter has detailed the use of the five commands and two routines available using the SEL 86/LSI-11 interface.

III. Operating Procedures

Introduction

The SEL 86/LSI-11 interface is designed to require minimum effort on the part of the user. The user interacts directly with the LSI-11. With the exception of login requirements, all interface activities between the SEL 86 and the LSI-11 are automatically monitored by the LSI-11 IM software. This allows a user unfamiliar with SEL interactive procedures to successfully use the SEL 86/LSI-11 interface.

This chapter specifies the procedures for accessing the SEL 86/LSI-11 interface. It consists of activating the LSI-11, getting into a transparency mode with the SEL 86, and logging in. At this point the user is free to use any of the functions available to the interface.

Procedure for Using SEL 86/LSI-11 Interface

1. Ensure that all needed equipment is available:

LSI-11

Keyboard

Printer or CRT

Modem

Telephone

Operational SEL

2. Turn on the LSI-11. The LSI-11 automatically jumps to the IM.

Response (from LSI-11): - -

3. Get into transparency mode in order to log in on the SEL 86:

Enter: TRANS <CR>

4. Dial the appropriate number to access a 300-baud rate line on the SEL 86.

Response (from SEL 86): - SEL TERMINAL SUPPORT
SYSTEM, TERMINAL xx-
ENTER USER NAME:

5. Enter an approved, validated name for the SEL 86 interactive mode.

Response (from SEL 86): ENTER USER KEY:

6. Enter: <CR>

Response (from SEL 86): ENTER FUNCTION CODE, 7,
OR OR TO TERMINATER

7. At this point, log in is complete on the SEL 86. The LSI-11 is still in a transparency mode. The user may continue in this mode or exit the transparency mode. To exit,

Enter: Control/D <CR>

Response (from LSI-11): - -

The user may now use any of the available commands of the SEL 86/LSI-11 interface.

Considerations in Using the Interface

Whenever the LSI-11 IM is awaiting keyboard input, it notifies the user with the prompting message:

- -

The SEL 86 will automatically log out the LSI-11 if the user or the LSI-11 IM does not interact with the SEL 86 for 30 minutes.

No specific keyboard input is required to terminate the LSI-11 IM. The LSI-11 IM is automatically terminated when the LSI-11 is turned off.

Summary

This chapter has specified the required steps in initializing the SEL 86/LSI-11 interface. The procedure is simply the standard procedure for logging onto the SEL 86 from a terminal once the LSI-11 has been placed into a transparency mode. This chapter does not go into any detail in accessing and using the SEL 86 interactive functions and procedures. The SEL Terminal Support Subsystem Reference Manual should be referenced for further detail in operating under the SEL 86 interactive mode.

Bibliography

1. Digital Microcomputer Handbook. 1977-78 Second edition. Maynard, Mass.: Digital Equipment Corporation, 1976.
2. SEL Terminal Support Subsystem Reference Manual. Ft. Lauderdale: Systems Engineering Laboratories, August 1977.

Appendix A

Absolute Load Format

A block consists of:

byte	value	
1	001	start byte
2	000	null byte
3	XXX	byte count (low 8 bits)
4	XXX	byte count (high 8 bits)
5	YYY	load address (low 8 bits)
6	YYY	load address (high 8 bits)
⋮	⋮	
n	ZZZ	block checksum

Notes:

1. The byte count is the total number of bytes in the block, excluding the checksum.
2. A byte count of six has specific implications:
 - a. data transfer is complete; and
 - b. the load address specifies the transfer address.

Appendix B

Bubble Charts for SEL 86/LSI-11 Interface Monitor

Five bubble charts are included in this appendix. They are: LSI-11 IM, LOAD, SEND, TRANS, and SEL 86 LSINTR. The bubble charts for FILE, RUN, IMCHAR, and IMEXIT have been excluded since these charts are trivial by the nature of what the modules must do. It should be noted before reading these charts that each chart represents data flow of a particular function. Control flow is intentionally omitted. The chart represents a series of data transformations from one form to another form as viewed by the data. This means that iterative loops and initialization procedures are not included in the charts.

The charts included here are only high level bubble charts. Each process, i.e. "bubble", could be decomposed into a lower level bubble chart to include more detail. The intent of these charts is simply to provide the reader with an overall concept of what is being done in each module before examining how the module is being implemented in the structure charts.

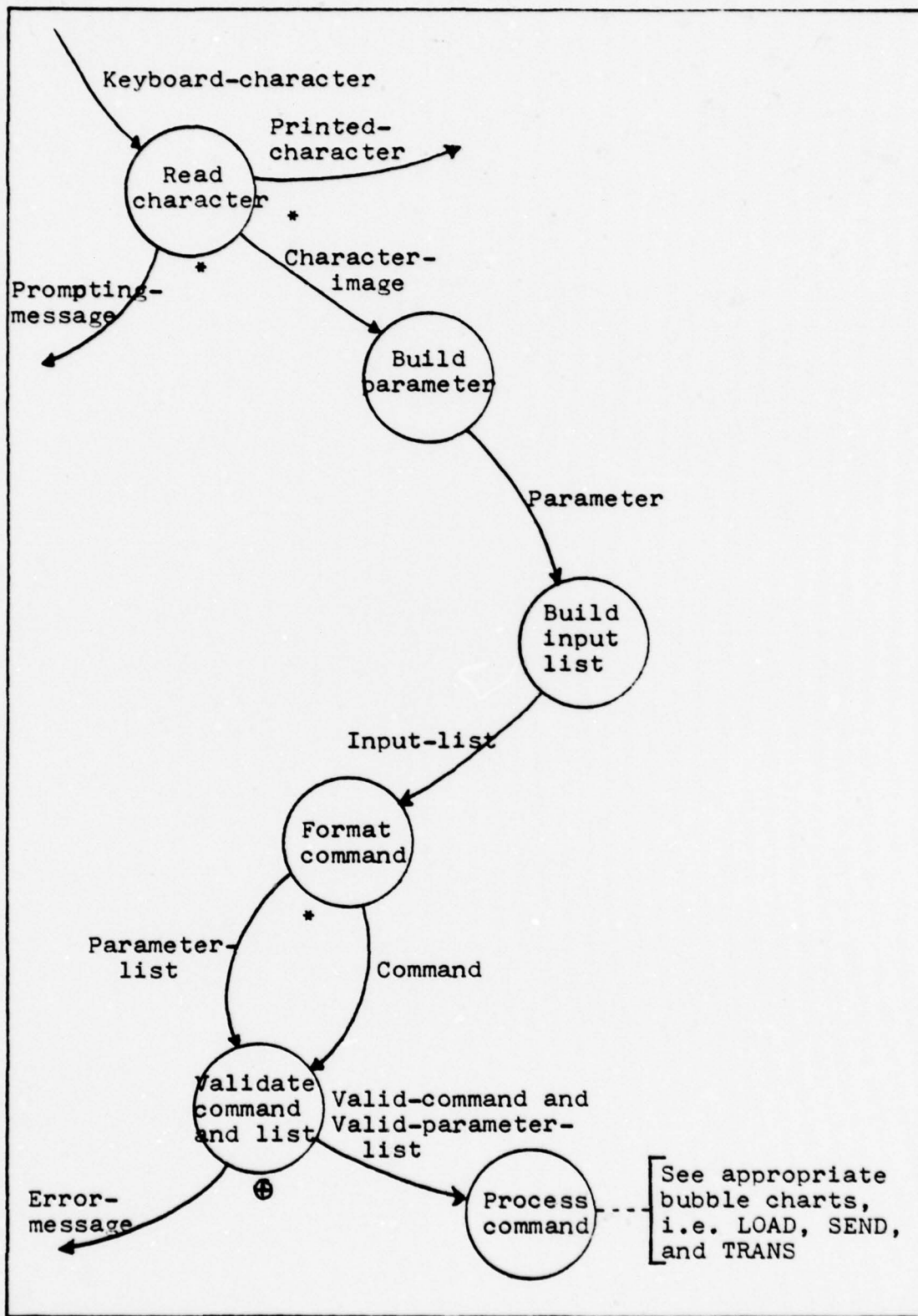


Figure 1. LSI-11 Interface Monitor Bubble Chart

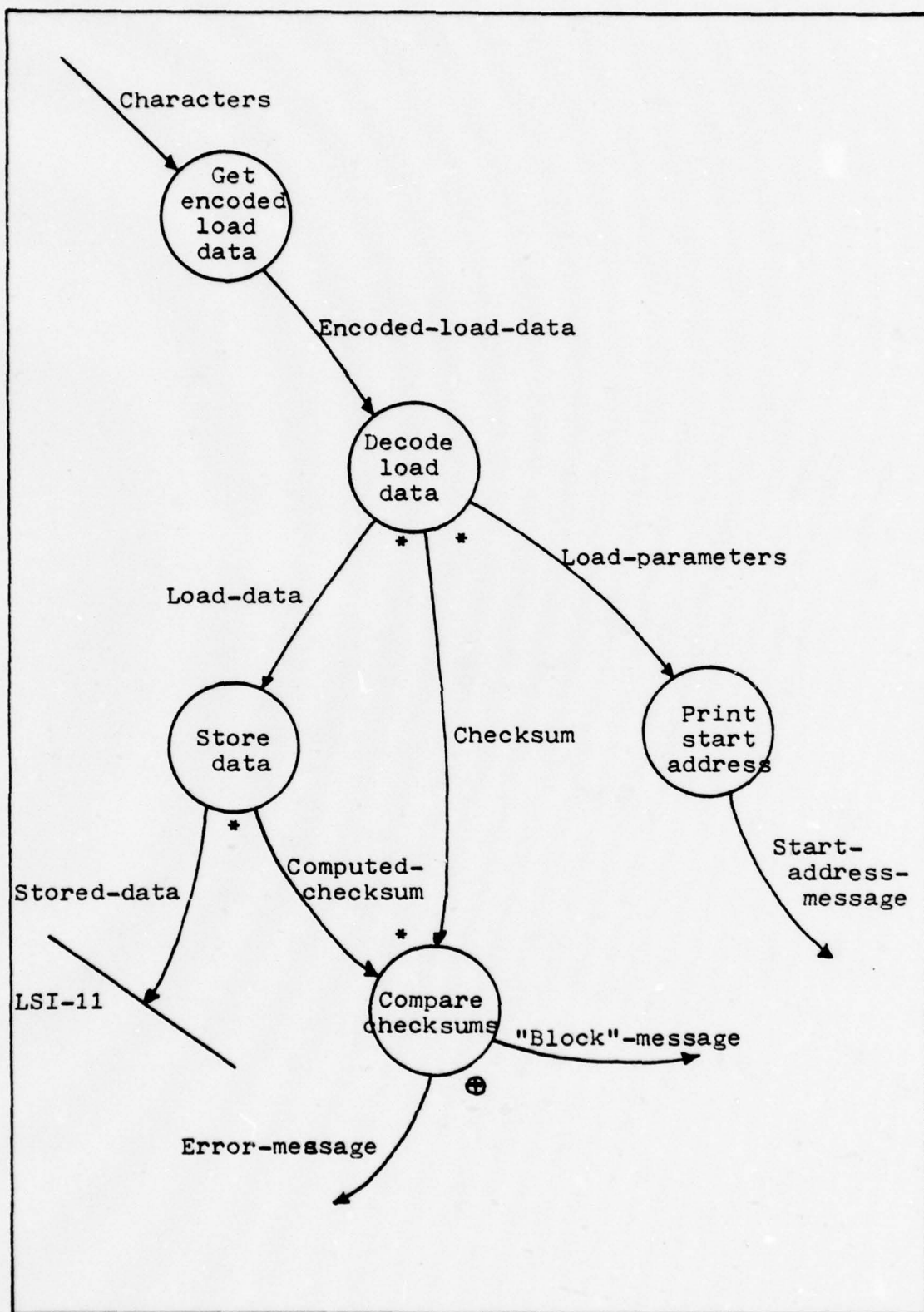


Figure 2. LOAD Bubble Chart

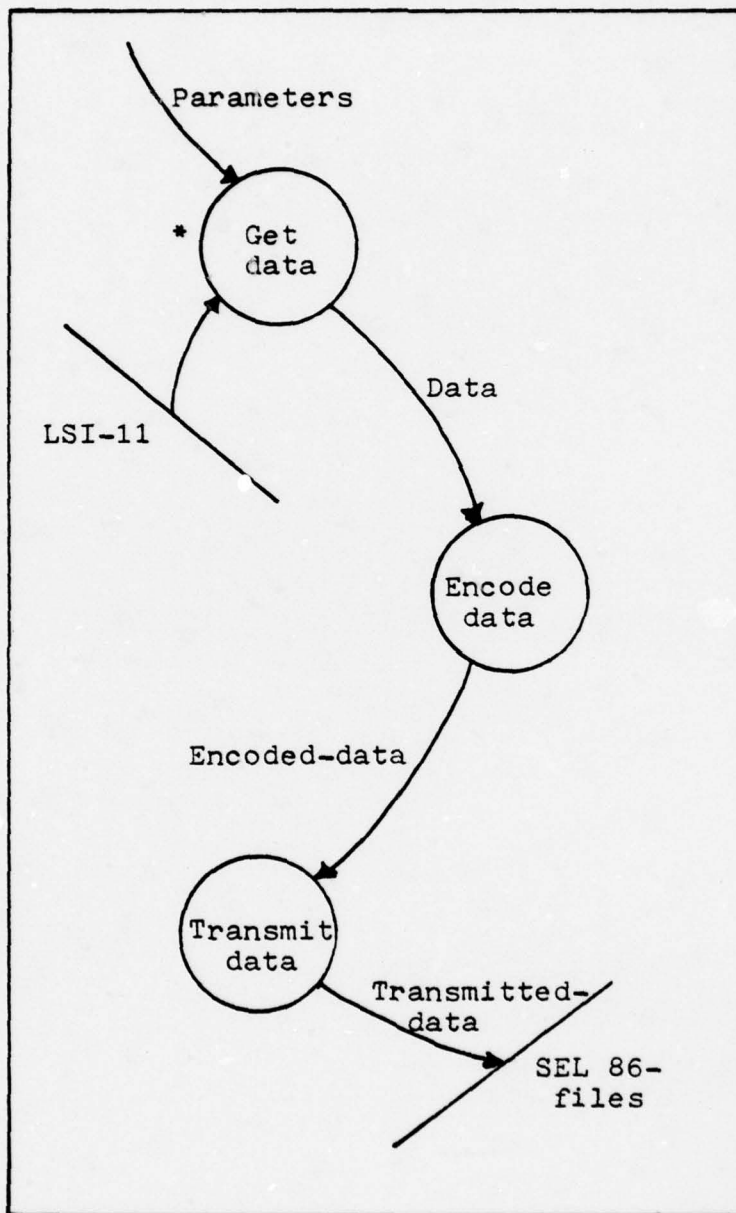


Figure 3. SEND Bubble Chart

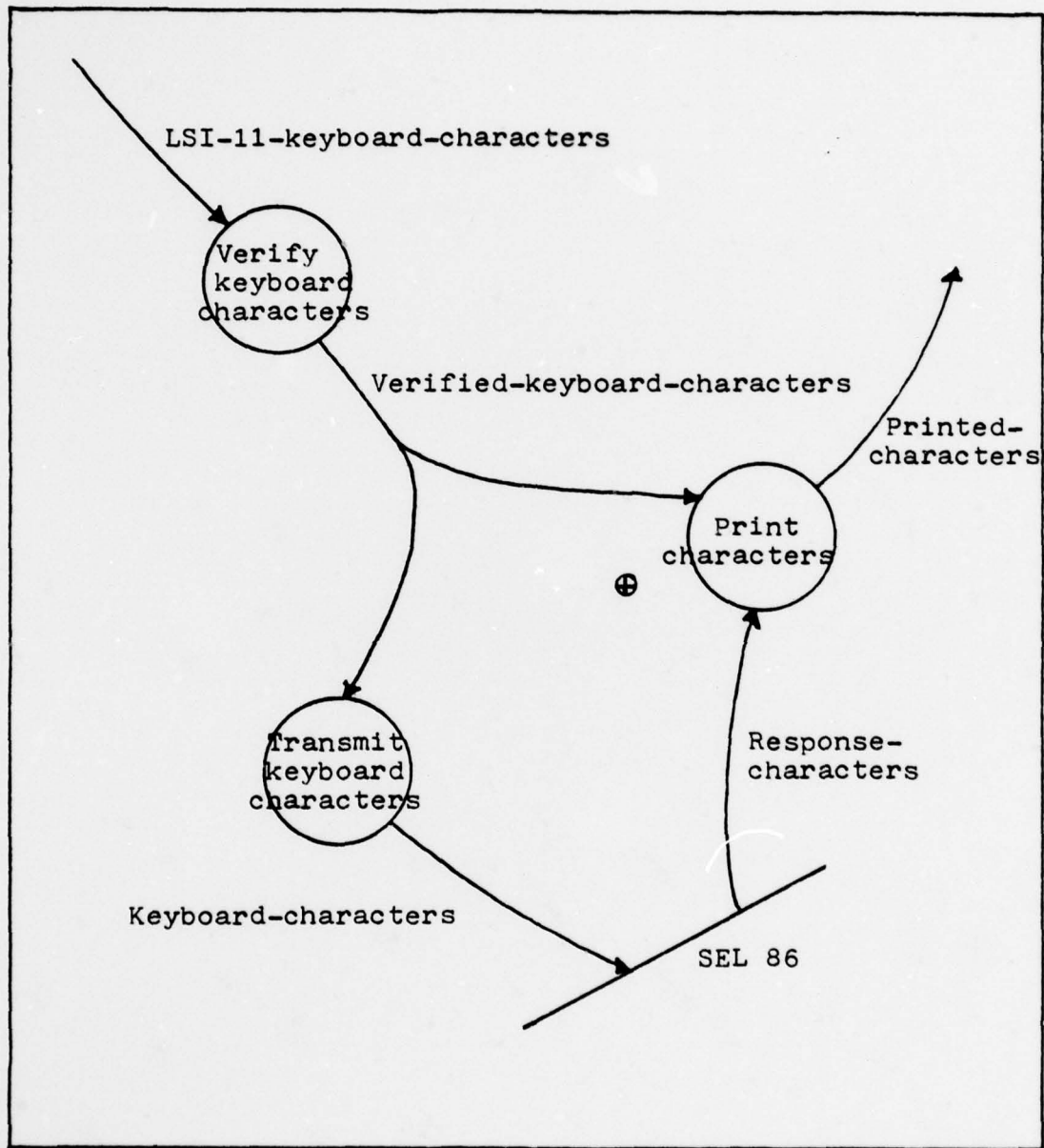


Figure 4. TRANS Bubble Chart

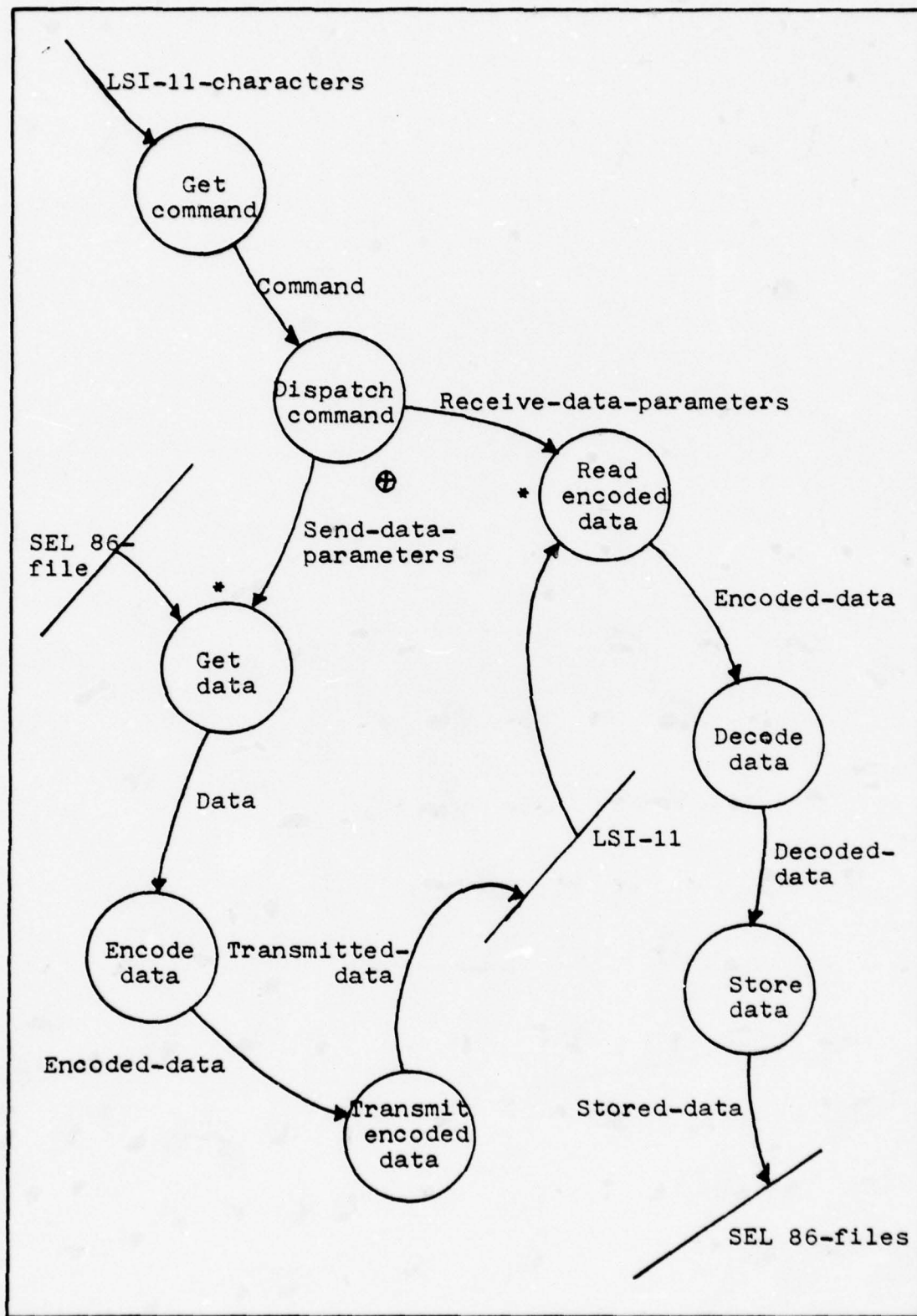


Figure 5. SEL 86 LSINTR Bubble Chart

Appendix C

Structure Charts for SEL 86/LSI-11

Interface Monitor

A series of eight structure charts represent the design of the SEL 86/LSI-11 Interface Monitor: LSI-IM Interface Monitor, LOAD, FILE/SEND, RUN, TRANS, IMCHAR, IMEXIT, and SEL 86 LSINTR. Each structure chart is followed by a table which specifies data and control flow parameters to (INPUT) and from (OUTPUT) the subroutines activated by the calling routine. Data and control flow are differentiated by underlining control flow parameters. Following this table is a description of all the modules specified in the structure chart.

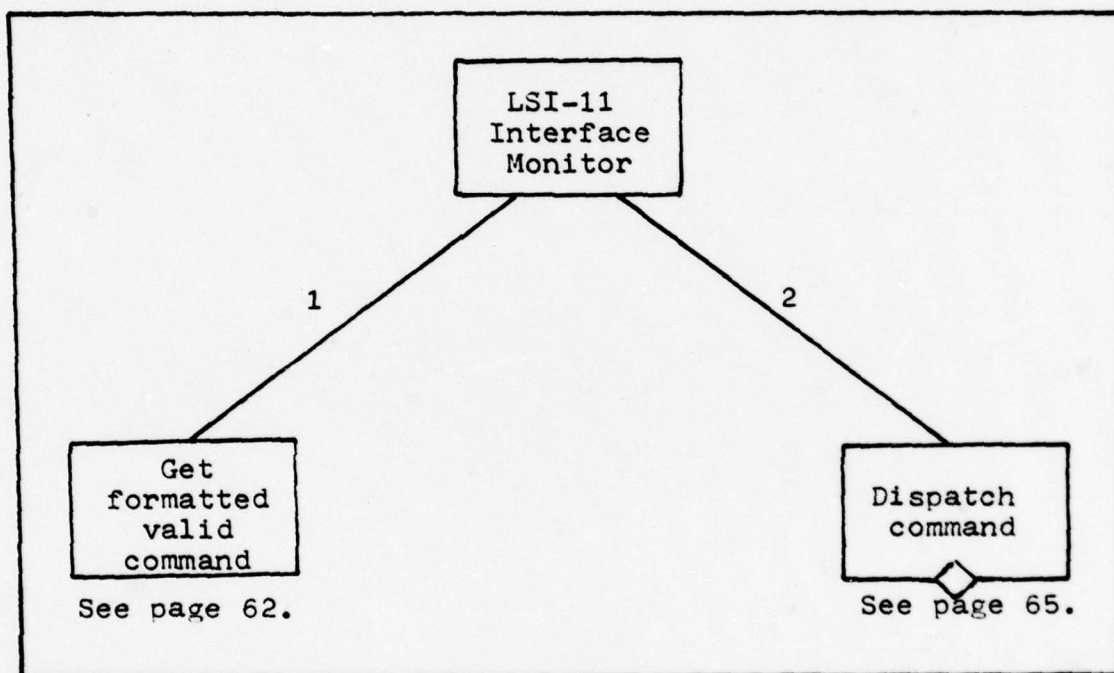


Figure 6. LSI-11 Interface Monitor Structure Chart

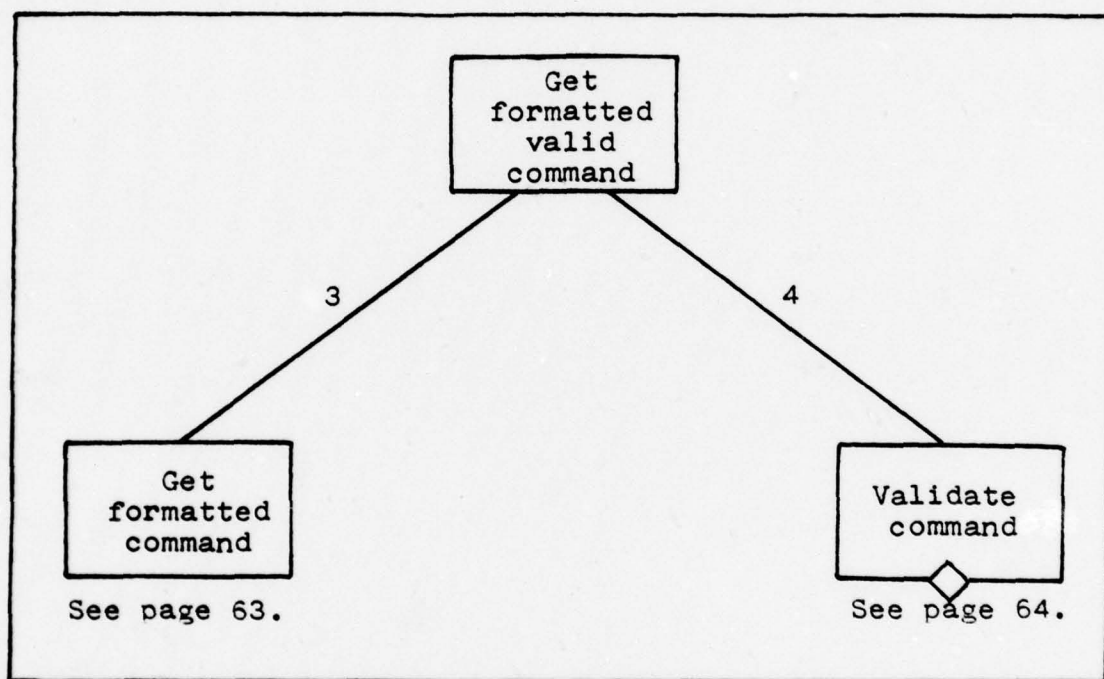


Figure 6. (cont'd)

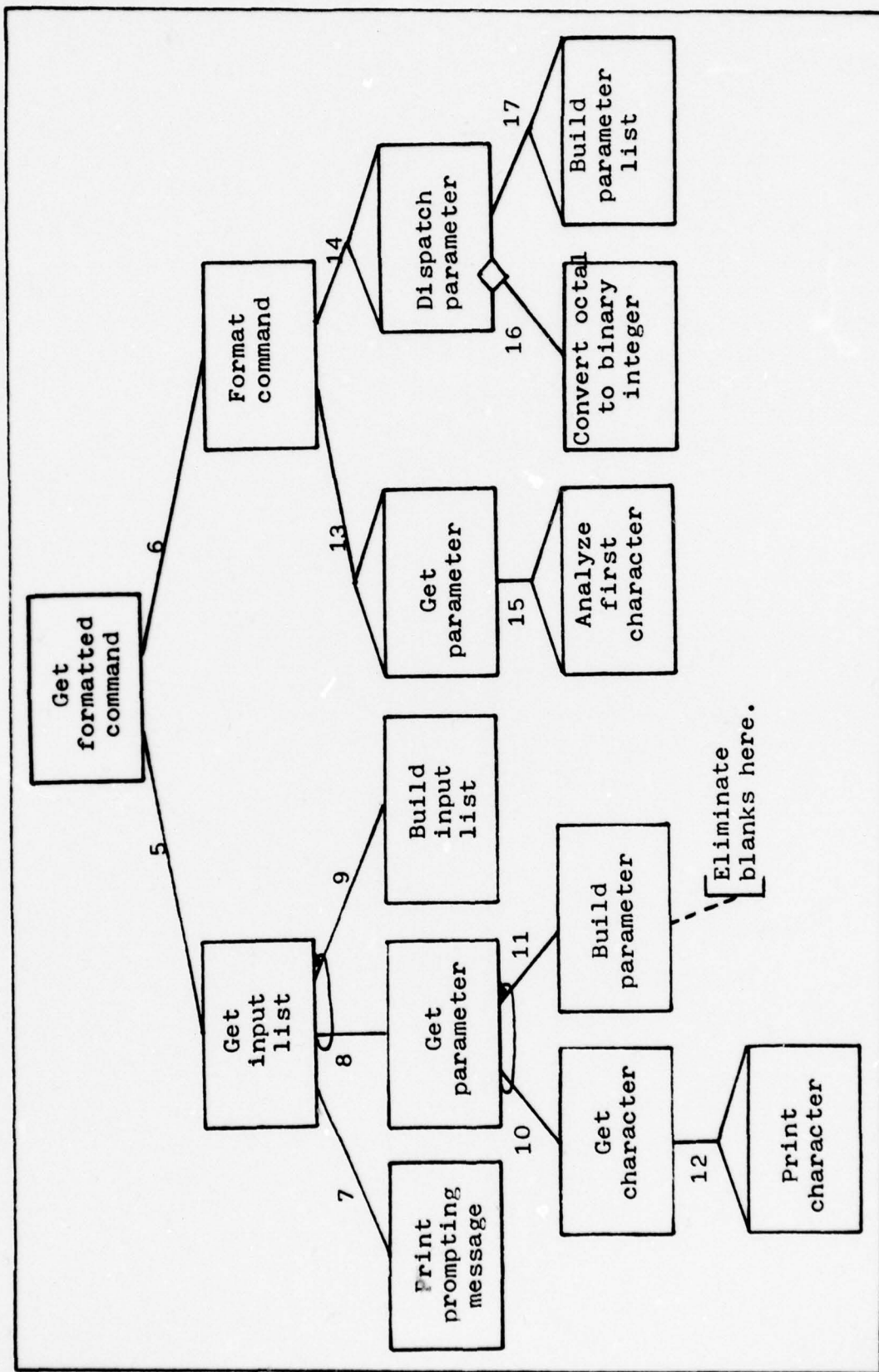


Figure 6. (cont'd)

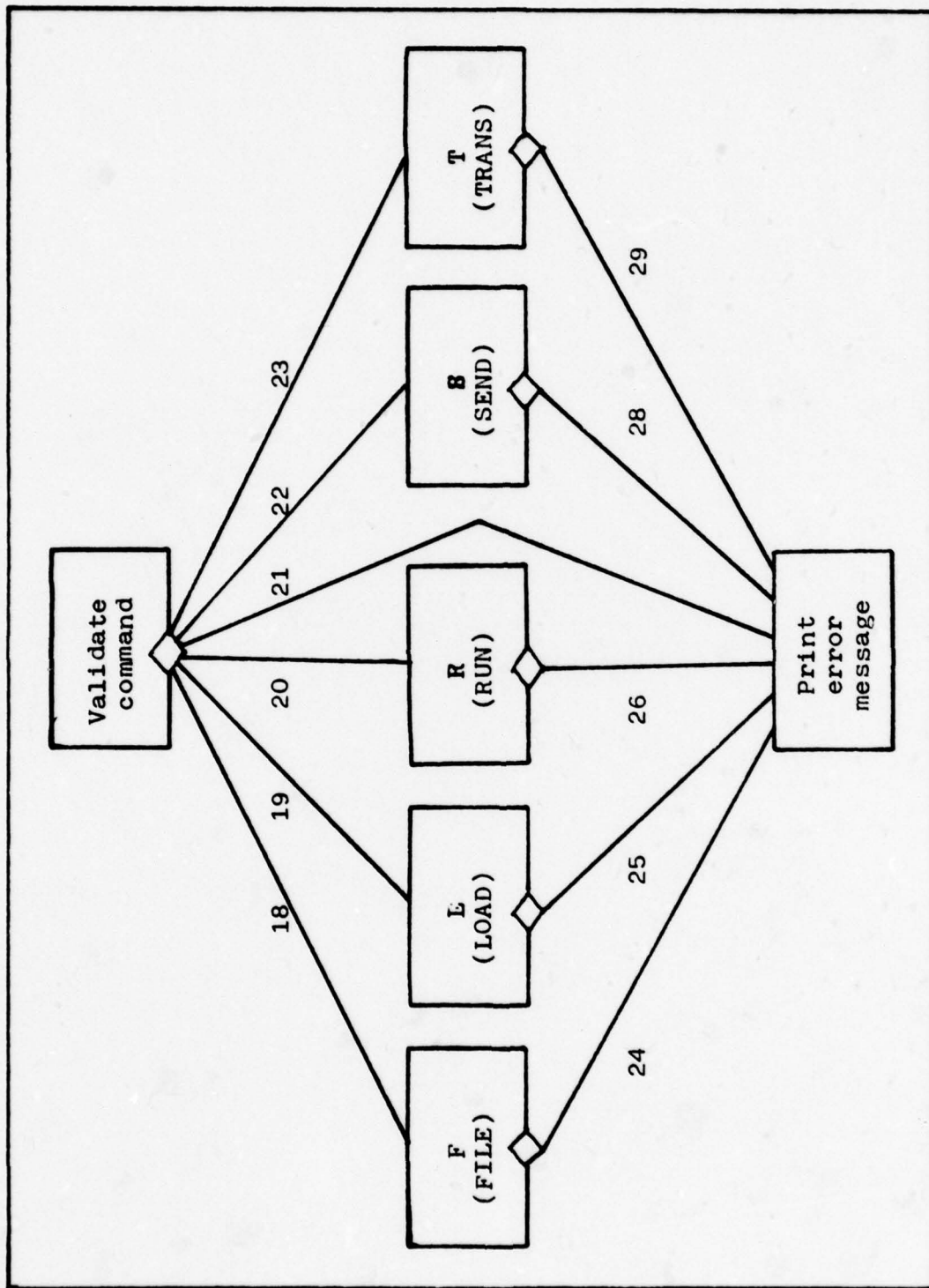


Figure 6. (cont'd)

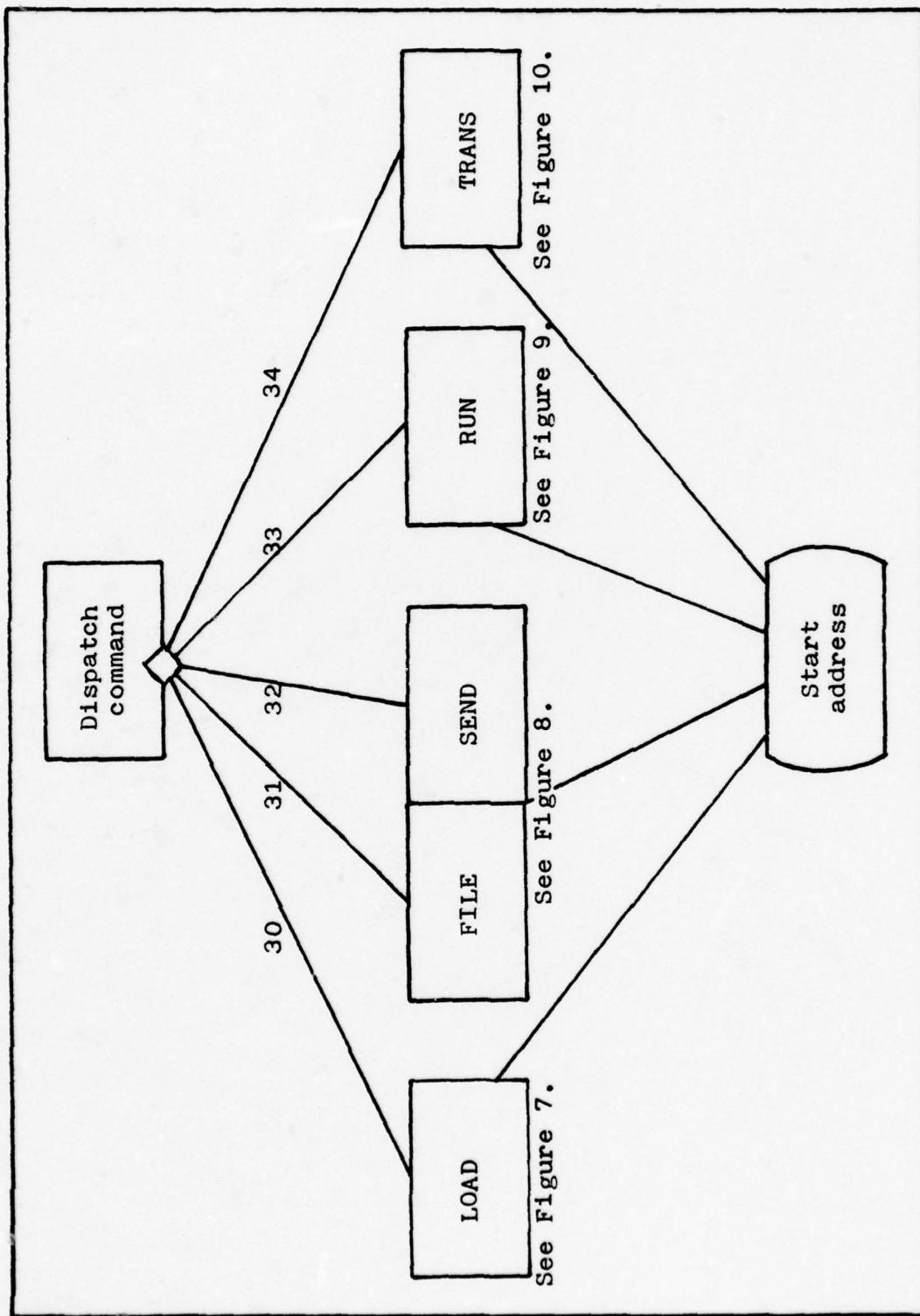


Figure 6. (cont'd)

Table I
LSI-11 Interface Monitor
Data and Control Flow

	INPUT	OUTPUT
1	-	command, parameter list
2	command, parameter list	-
3	-	command, parameter list
4	commnad, parameter list	command type
5	-	input list
6	input list	command, parameter list
7	-	-
8	-	parameter
9	parameter	input list
10	-	character
11	character	parameter
12	character	-
13	input list	parameter type
14	parameter type	parameter list
15	parameter	parameter type
16	numeric parameter	integer parameter
17	parameter	parameter list
18-23	command, parameter list	command type
24-29	-	-
30-34	parameter list	-

Description of LSI-11 Interface Monitor

LSI-11 IM - controls procedure of getting a valid command and then dispatching the command for processing.

Get formatted valid command - gets a command and a parameter list (FORTRAN format) and then validates them. When deciding on validity of command and list, determines command type (i.e. FILE command, LOAD command, etc). If command is not valid, gets another command.

Dispatch command - transfers the parameter list to the appropriate routine to process the command.

Get formatted command - gets a command and its parameters and builds a parameter list in FORTRAN format.

Validate command - reads first ASC11 character of the command parameter to dispatch the command and parameter list to determine their validity. If character is not one of five valid characters. Prints error message and returns invalid command type.

Get input list - prints a prompting message to the user and builds an input list from keyboard input. Keyboard entry is complete when no parameter is found.

Format command - converts input list to command and parameter list.

Print prompting message - prints "--" at the user's terminal.

Get parameter - reads ASC11 characters from keyboard input and builds parameters. A parameter is built when the ASC11 character read is a comma or carriage return.

Build input list - adds parameter to the input list.

Get character - reads ASC11 character from keyboard input and prints the character to the LSI-11 user's console.

Build parameter - builds an ASC11 character string of the parameter. Blanks input as ASC11 characters are deleted from the ASC11 parameter string.

Print character - echo print user's keyboard character entry at the user's LSI-11 console.

Get parameter - reads the next parameter in the input list. Determines parameter type by reading first ASC11 character of the parameter and noting if it is an alphanumeric or a numeric.

Dispatch parameter - converts numeric parameter to binary integer if necessary. Adds parameter to FORTRAN formatted parameter list.

Analyze first character - determine if first character of the parameter is an alphanumeric or a numeric.

Convert octal to binary integer - converts input octal integer parameter to binary integer parameter.

Build parameter list - adds parameter to FORTRAN formatted parameter list.

F,L,R,S,T - validates the command and parameter list as either FILE, LOAD, RUN, SEND, or TRANS command. Determines command type. If an error in command or parameters, prints an error message and returns invalid command type.

Pring error message - prints error message to user's LSI-11 console that error in input command.

LOAD, FILE/SEND, RUN, TRANS- see individual structure charts.

Start address-only contents of this module is the start address. LOAD, FILE/SEND, RUN, and TRANS all have access to this common data element.

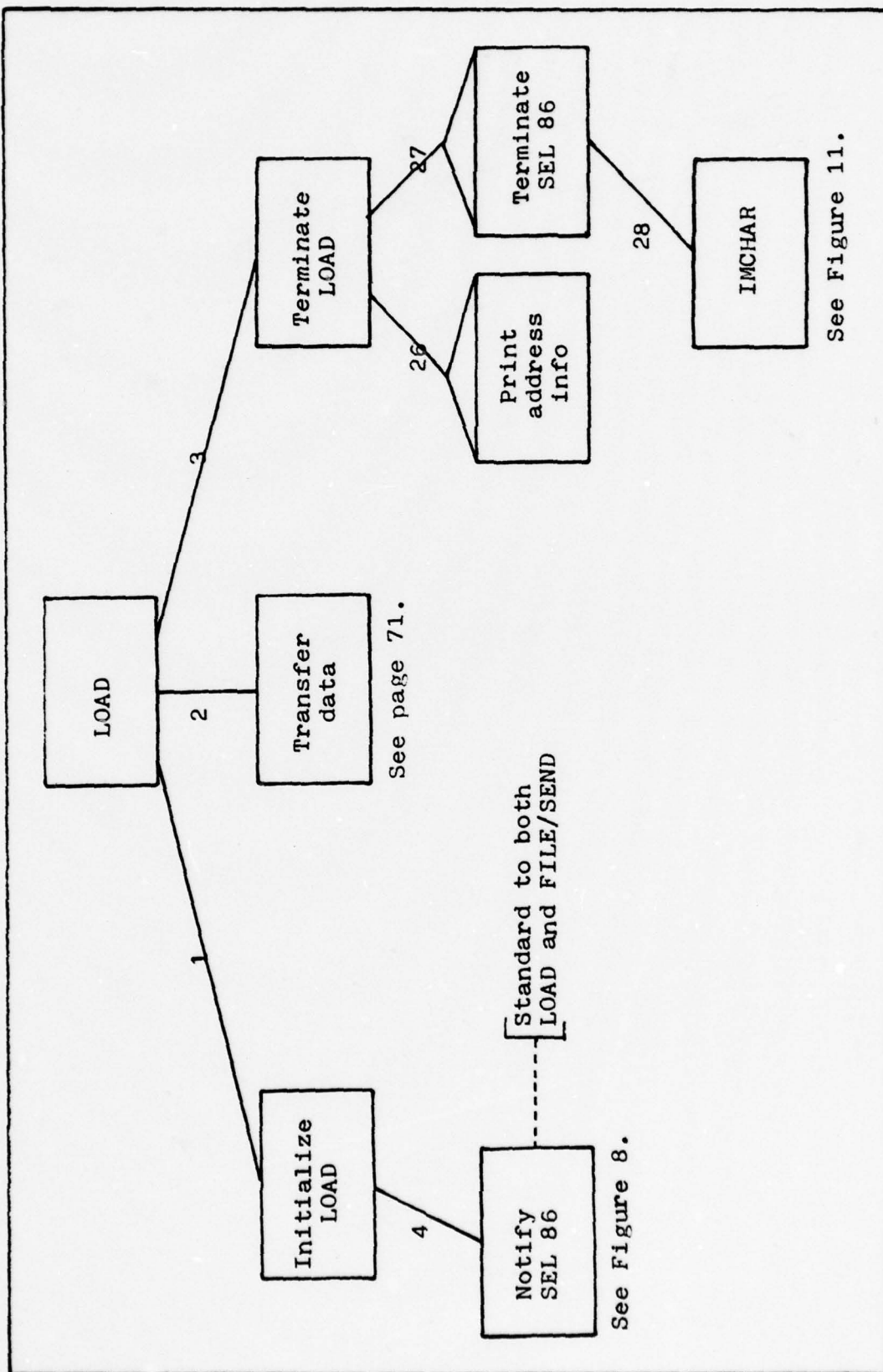
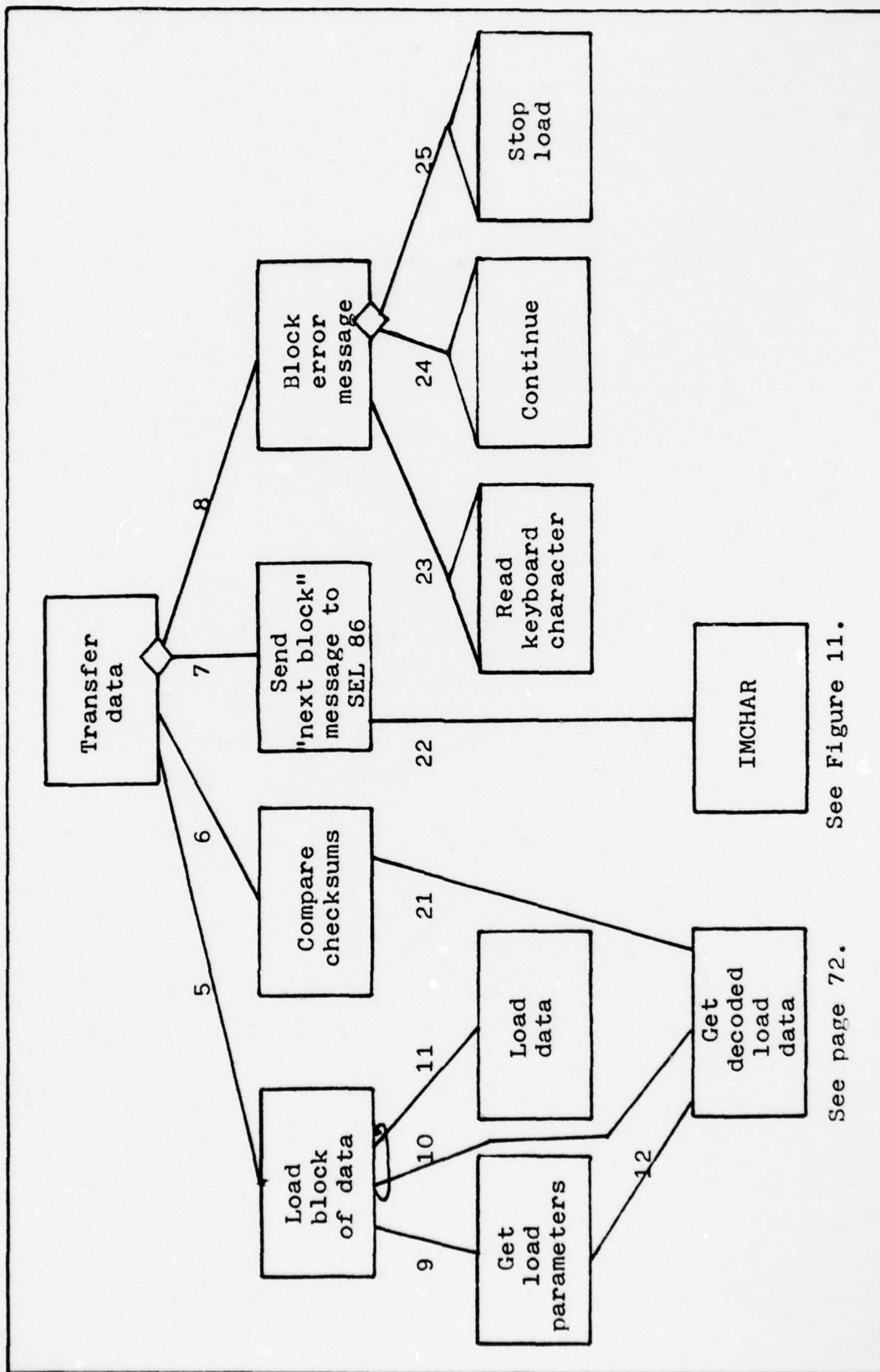


Figure 7. LOAD Structure Chart



See Figure 11.

See page 72.

Figure 7. (cont'd)

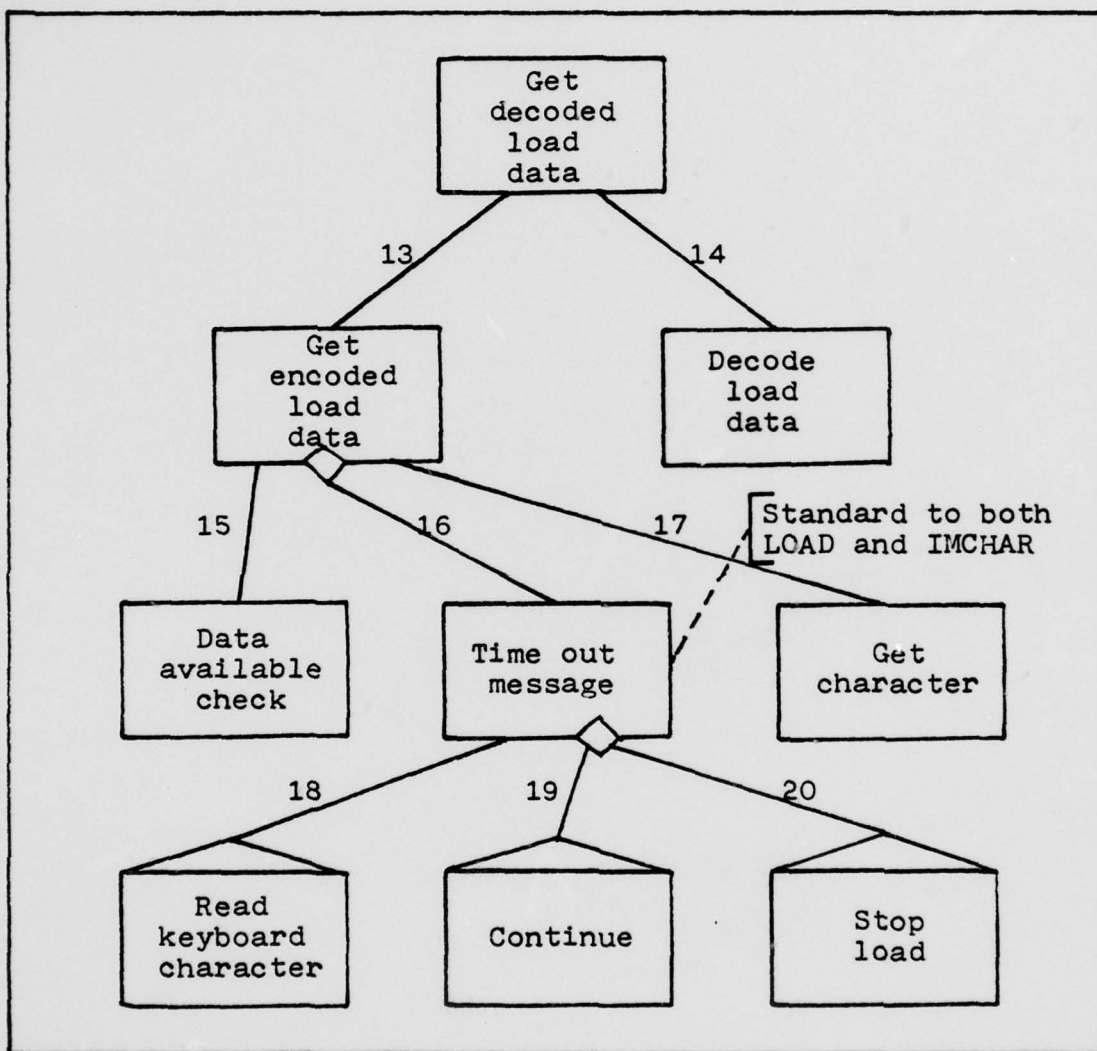


Figure 7. (cont'd)

Table II
LOAD Data and
Control Flow Table

	INPUT	OUTPUT
1	parameter list	load offset
2	load of set	-
3	start address	-
4	SEL 86 mode, file name	-
5	-	checksum, block number <u>stop flag</u>
6	checksum	<u>checksum flag</u> , <u>stop flag</u>
7	-	-
8	block number	<u>stop flag</u>
9	-	load address, number of bytes, <u>stop flag</u>
10	-	decoded data, <u>stop flag</u>
11	load address, number of bytes, decoded data	updated load address, updated number of bytes
12	-	decoded load parameter, <u>stop flag</u>
13	-	encoded load data, <u>stop flag</u>
14	encoded load data	decoded load data
15	-	-
16	-	<u>stop flag</u>
17	-	encoded character
18	-	keyboard ASCII character
19	-	-

Table II (continued).

	INPUT	OUTPUT
20	-	<u>stop flag</u>
21	-	load checksum, <u>stop flag</u>
22	character string, number of characters	-
23	-	keyboard ASC11 character
24	-	-
25	-	<u>stop flag</u>
26	start address	-
27	-	-
28	character string, number of characters	-

Description of LOAD

LOAD - controls load of data from SEL 86 file to LSI-11 memory.

Initialize load - determines parameters for notifying SEL 86. Determines load offset. If not specified, defaults to zero.

Notify SEL 86 - control initialization of SEL 86 LSINTR and specification of SEL 86 parameters, i.e. mode and file name.

Initialize SEL 86 LSINTR - initialize SEL 86 LSINTR by putting SEL 86 in Program Monitor and executing LSINTR. Notifies SEL 86 that it is in sending mode.

IMCHAR - see IMCHAR structure chart.

Transfer data - controls receipt of load data from SEL 86 and transfer of the data into LSI-11 memory. Monitors two control flags: stop flag and checksum flag. If stop flag is set, load is terminated. If checksum flag is set, block error message must be printed.

Load block of data - controls receipt of load data from SEL 86 and transfer of the data into LSI-11 memory. Keeps track of block number of the load. Keeps running tally of checksum. Monitors stop flag. If set, terminates load.

Get load parameters - determines load address and number of bytes as specified in load parameters of the block being transmitted.

Get decoded load data - controls receipt of decoded load data. Monitors stop flag.

Get encoded load data - controls receipt of encoded load data from the SEL 86. Monitors stop flag.

Decode load data - decodes ASC11 characters (encoded load data) into binary data.

Data available check - loop of executing statements to count length of time waiting for data from SEL 86.

Time-out message - prints message specifying allotted time for waiting for data from SEL 86 has run out. Controls option to continue waiting for data or to terminate load.

Get character - reads ASC11 character from SEL 86.

Read keyboard character - reads LSI-11 keyboard input character specifying continue waiting or terminate load.

Continue - continue waiting.

Stop load - sets stop flag.

Load data - controls load address to store data into LSI-11 memory. Decrements byte count. Loads data into LSI-11 memory.

Compare checksums - gets checksum as transmitted in load block and compares it to the calculated checksum. If a discrepancy, checksum flag is set.

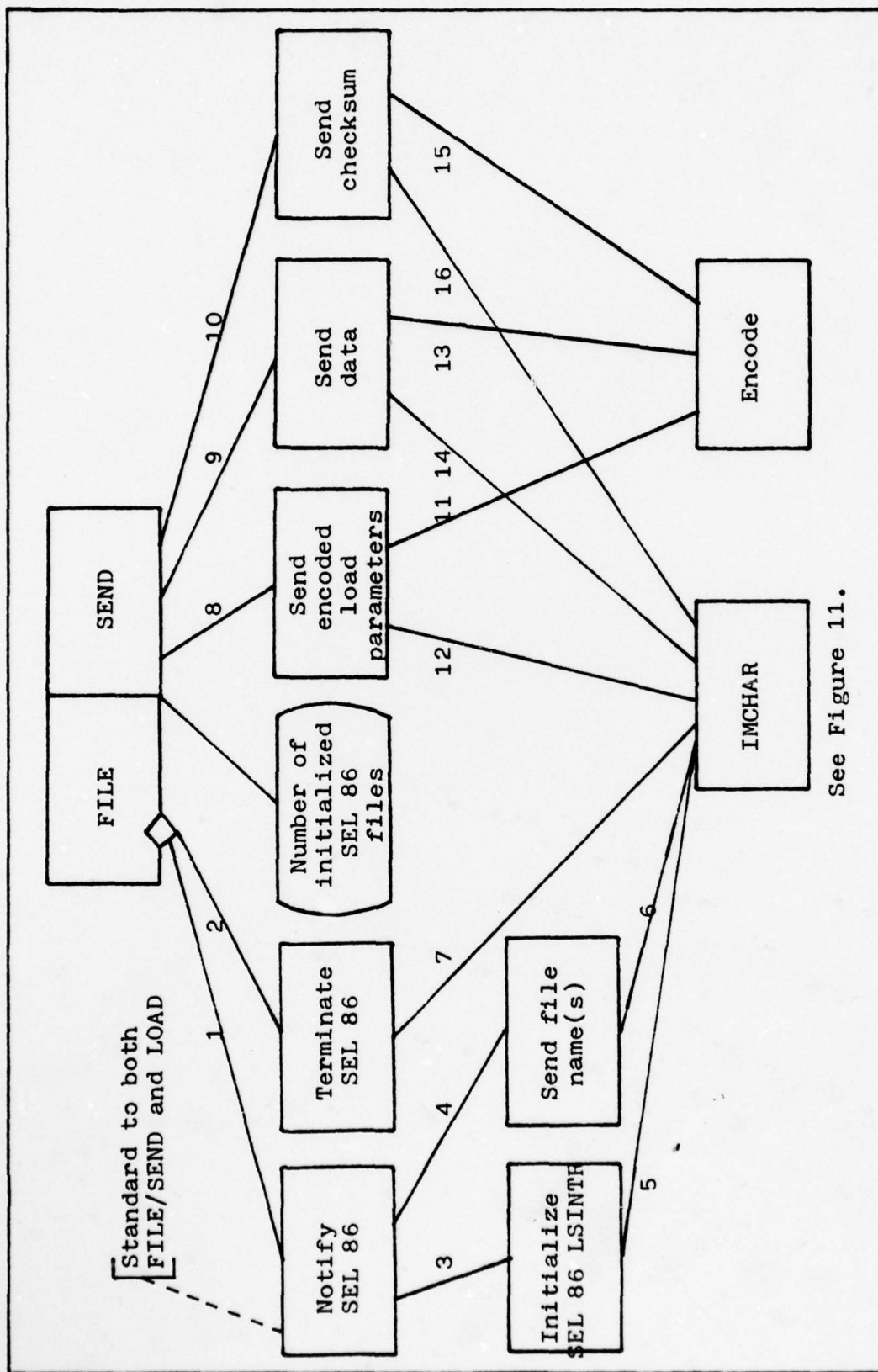
Send "next block" message to SEL 86 - sends notification to SEL 86 to start transmission of next block of data.

Block error message - prints block error message (i.e. checksum error). Gives user option to continue or terminate load as in "Time-out message".

Print address information - prints start address as specified by the load.

Terminate SEL 86 - terminates data receipt from SEL 86 by stopping execution of SEL 86 LSINTR. Terminates Program Monitor mode on SEL 86.

IMCHAR - see IMCHAR structure chart.



See Figure 11.

Figure 8. FILE/SEND Structure Chart

Table III
FILE/SEND Data and
Control Flow Table

	IN	OUT
1	SEL 86 mode, file name(s)	-
2	-	-
3	SEL 86 mode	-
4	file name(s)	-
5-7	character string, number of characters	-
8	parameter list	data address, number of bytes, checksum, <u>stop flag</u>
9	data address, number of bytes, checksum	checksum, <u>stop flag</u>
10	checksum	<u>stop flag</u>
11	load parameters	encoded load parameters
12	encoded load parameters, number of characters	<u>stop flag</u>
13	data	encoded data
14	encoded data, number of characters	<u>stop flag</u>
15	checksum	encoded, checksum
16	encoded checksum, number of characters	<u>stop flag</u>

Description of FILE/SEND

FILE - controls notification of SEL 86 to initialize its program, LSINTR, or to terminate it. If parameter list has zero entries, SEL 86 LSINTR is terminated, otherwise, the SEL 86 program is initialized. Also specifies that SEL 86 will be receiving data. Controls sending of file names to SEL 86. SEL 86 traffic to LSI-11 can interrupt routine. SEL 86 messages will be printed to LSI-11 user's console.

Notify SEL 86 - see LOAD structure chart and description.

Terminate SEL 86 LSINTR - terminates data transmission by stopping execution of SEL 86 LSINTR. Terminates Program Monitor mode on SEL 86.

IMCHAR - see IMCHAR structure chart

Process SEL 86 - see TRANS structure chart and description.

SEND - controls transmission of data in absolute load block format. Monitors number of files initialized on SEL 86. If this number is zero, transmission of data is zero. SEL 86 traffic to LSI-11 can interrupt routine. SEL 86 messages will be printed to LSI-11 user's console.

Send encoded load parameters - controls transmission of load parameters, i.e. file number, start address, and byte count. File number is sent using the state byte (high 8 bits) and null byte (low 8 bits) to specify file on

the SEL 86. Specifies number of ASC11 characters to be transmitted for each parameter. Keeps running tally of checksum count for encoded data transmitted to the SEL 86. Monitors stop flag (terminate transmission). If stop flag is set, number of initialized files is changed to zero.

Send data - controls transmission of the data. Specifies number of ASC11 characters to be transmitted for each data byte transmission. Keeps running tally of the checksum Monitors stop flag as does "Send encoded load parameters".

Send checksum - controls transmission of checksum. Specifies number of ASC11 characters to be transmitted. Monitors stop flag as does "Send encoded load parameters",

Encode - encodes each four bits of data into an ASC11 numeric character.

IMCHAR - see IMCHAR structure chart.

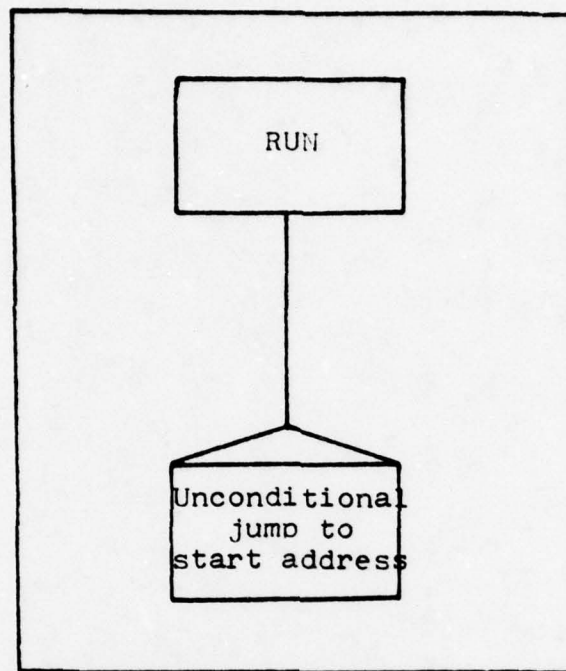


Figure 9. RUN Structure Chart

Table IV

RUN Data and
Control Flow Table

INPUT		OUTPUT
1	start address	-

Description of RUN

RUN - reads parameter list for start address. If none specified uses the start address data element common to all commands. Unconditionally jumps to that address.

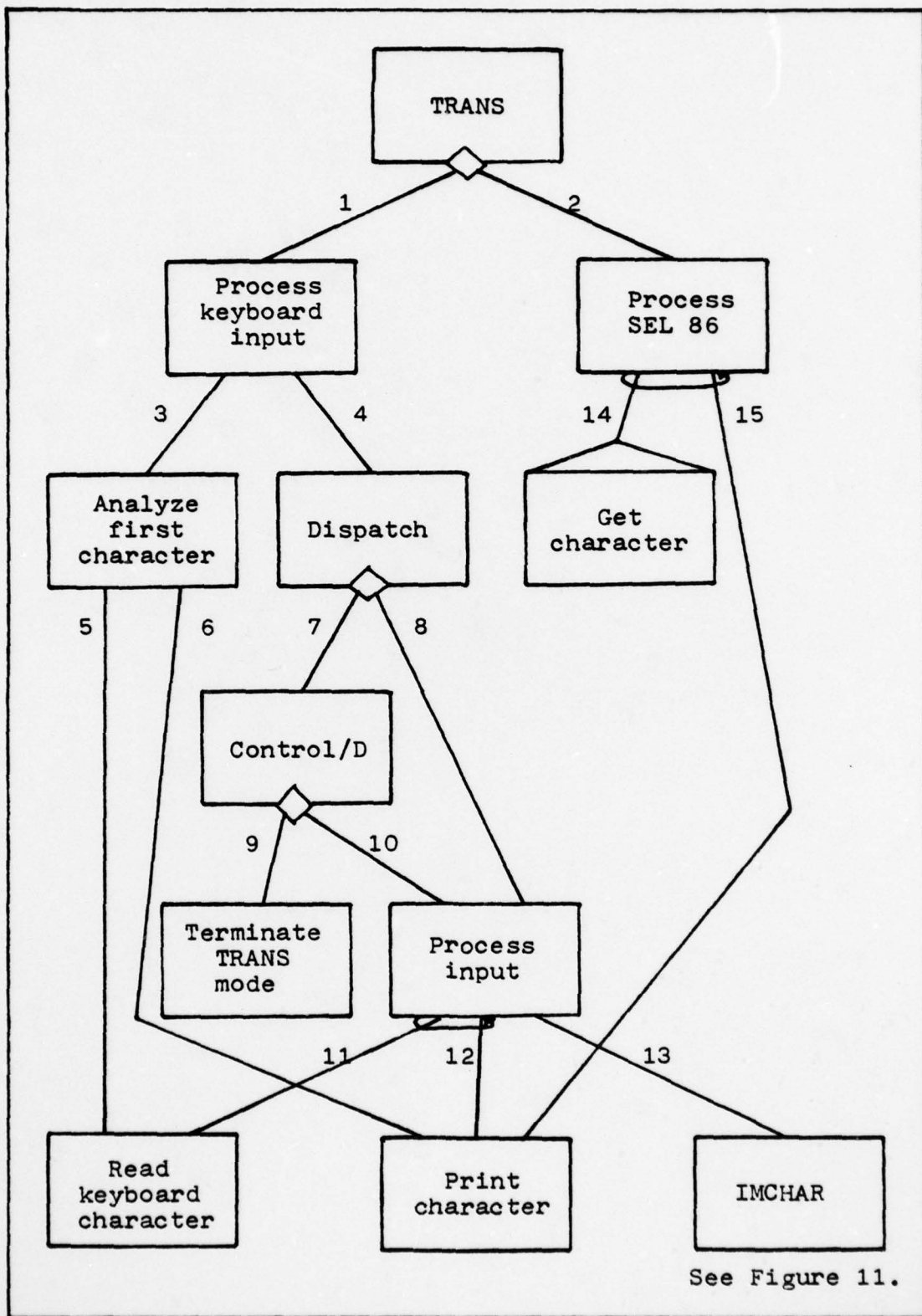


Figure 10. TRANS Structure Chart

Table V
TRANS Data and
Control Flow Table

	INPUT	OUTPUT
1	-	-
2	-	-
3	-	character
4	character	-
5	-	character
6	character	-
7	-	-
8	-	-
9	-	-
10	-	-
11	-	character
12	character	-
13	character string, number of characters	-
14	-	character
15	character	-

Description of TRANS

TRANS - controls processing of keyboard input or SEL 86 response. Determination is interrupt-driven.

Process keyboard input - monitors keyboard input to determine whether to remain in transparency mode or terminate transparency. Echo prints all keyboard input.

Process SEL 86 - echo prints SEL 86 ASC11 character response.

Analyze first character - reads and prints first LSI-11 keyboard character.

Dispatch - determines if valid transparency keyboard input and processes it or if termination of transparency mode request and processes request.

Control/D - if first keyboard character is control/D, verifies if transparency keyboard input or termination of transparency mode request verification is made by second ASC11 character, i.e. <CR>.

Terminate TRANS mode - return control to LSI-11 IM.

Process input - controls processing of keyboard input, i.e. echo prints and transmits to SEL 86. Terminates on <CR>.

Read keyboard character - reads character from LSI-11 keyboard.

Print character - echo print ASC11 character read.

IMCHAR - see IMCHAR structure chart.

Get character - read ASC11 character sent from SEL 86.

AD-A064 396

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO
DESIGN OF A SEL 86/LSI-11 INTERFACE MONITOR.(U)
DEC 78 J E BARALLI

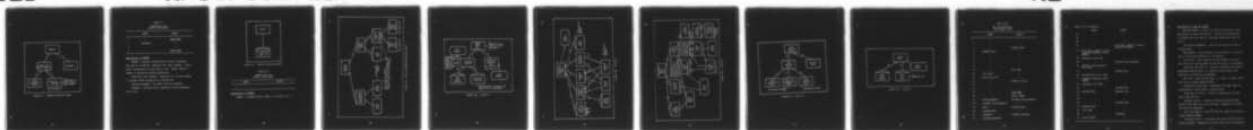
SCH--ETC F/G 9/2

UNCLASSIFIED

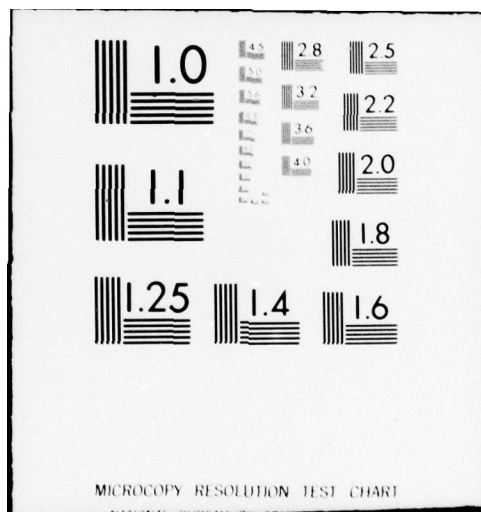
AFIT/GCS/EE/78-9

NL

2 OF 2
AD
A064396



END
DATE
FILMED
4-79
DDC



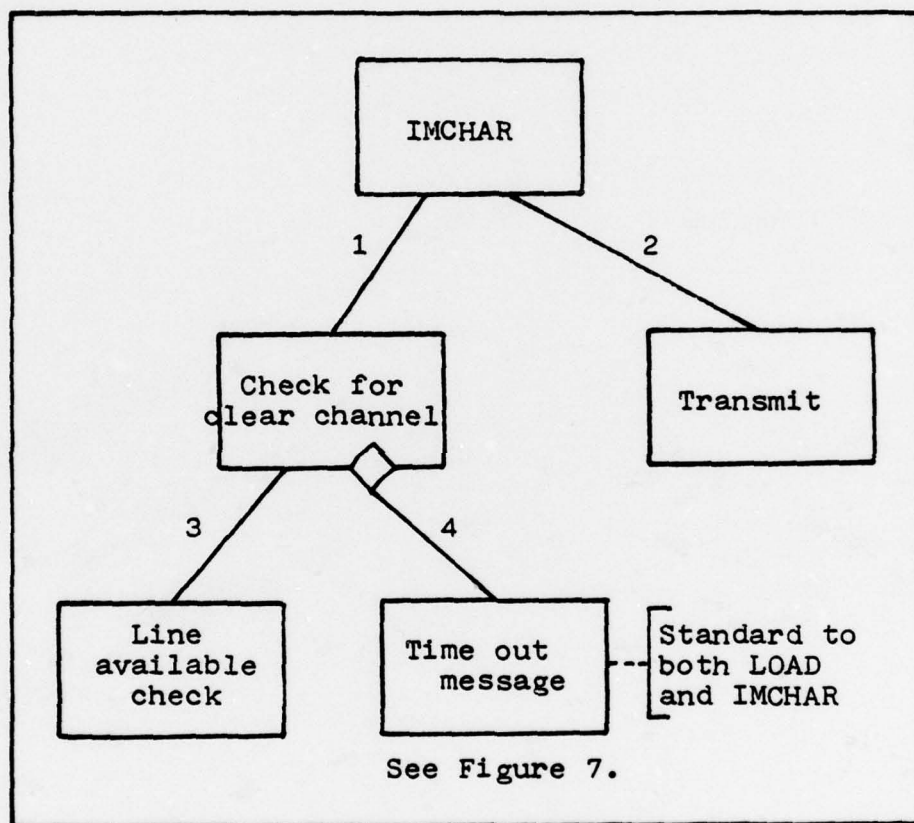


Figure 11. IMCHAR Structure Chart

Table VI
IMCHAR Data and
Control Flow Table

	INPUT	OUTPUT
1	-	<u>stop flag</u>
2	character	-
3	-	-
4	-	<u>stop flag</u>

Description of IMCHAR

IMCHAR - controls transmission of ASC11 characters from LSI-11 to SEL 86. Monitors for clear channel. Terminates transmission if stop flag is set. Keeps track of number of characters being transmitted.

Check for clear channel - monitors for a clear channel to transmit ASC11 characters to the SEL 86.

Time-out message - see LOAD "Time-out message".

Transmit - releases ASC11 character across communications line.

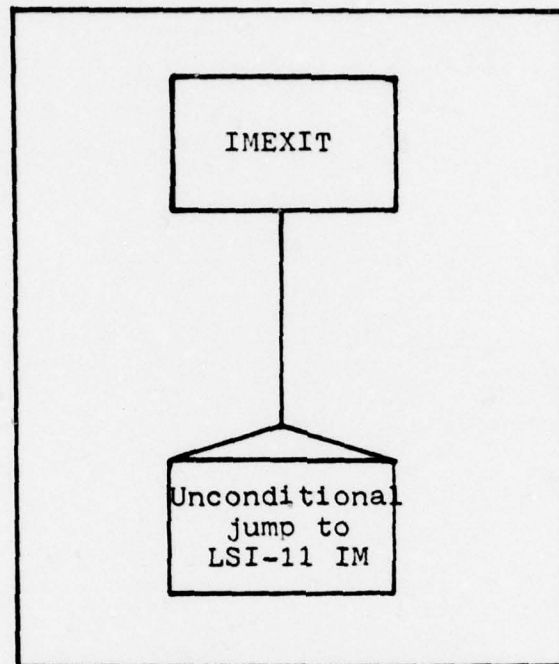


Figure 12. IMEXIT Structure Chart

Table VII

IMEXIT Data and
Control Flow Table

INPUT		OUTPUT
1	-	-

Description of IMEXIT

IMEXIT - unconditionally jumps to the LSI-11 IM

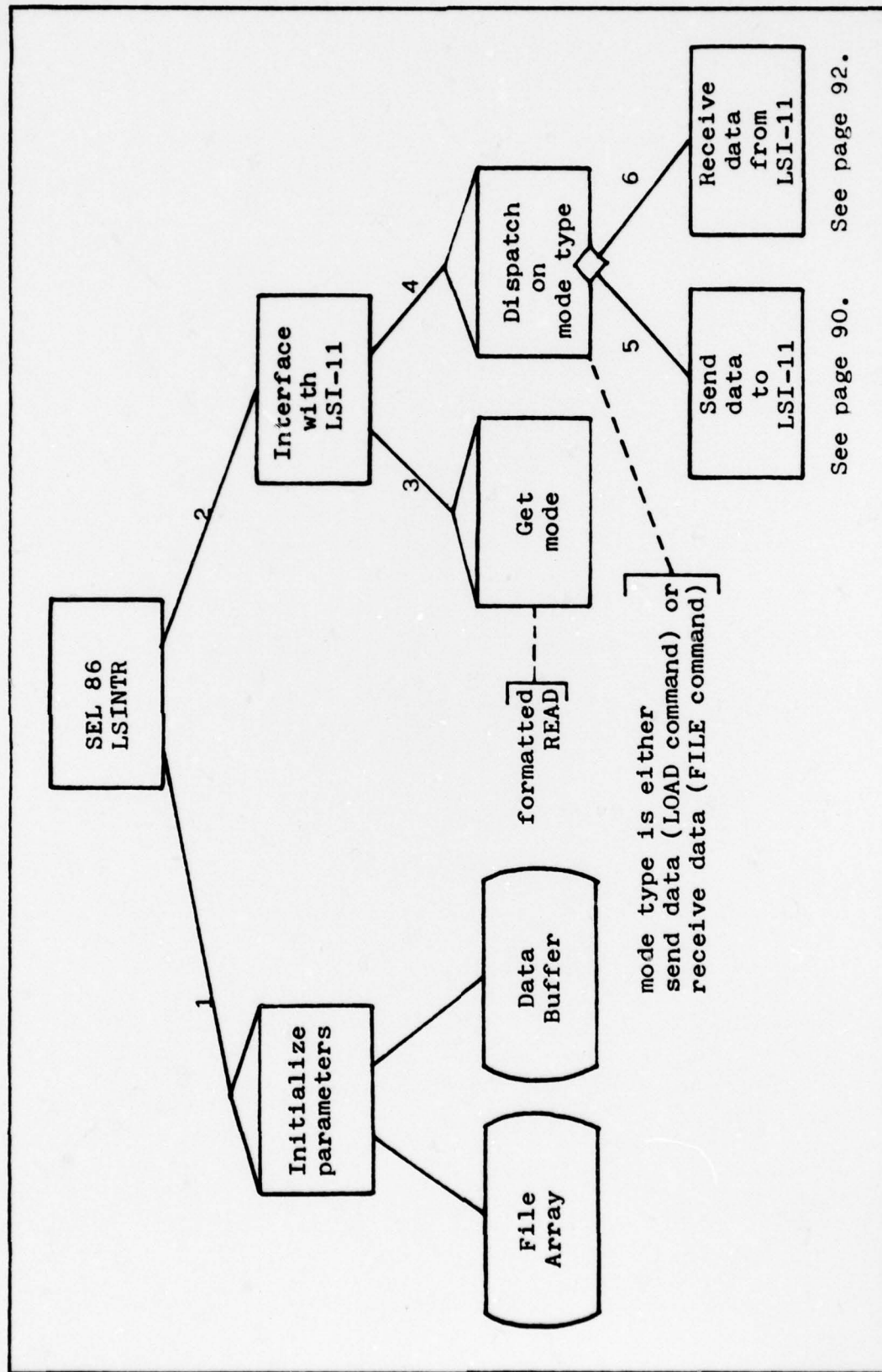


Figure 13. SEL 86 LSINTR Structure Chart

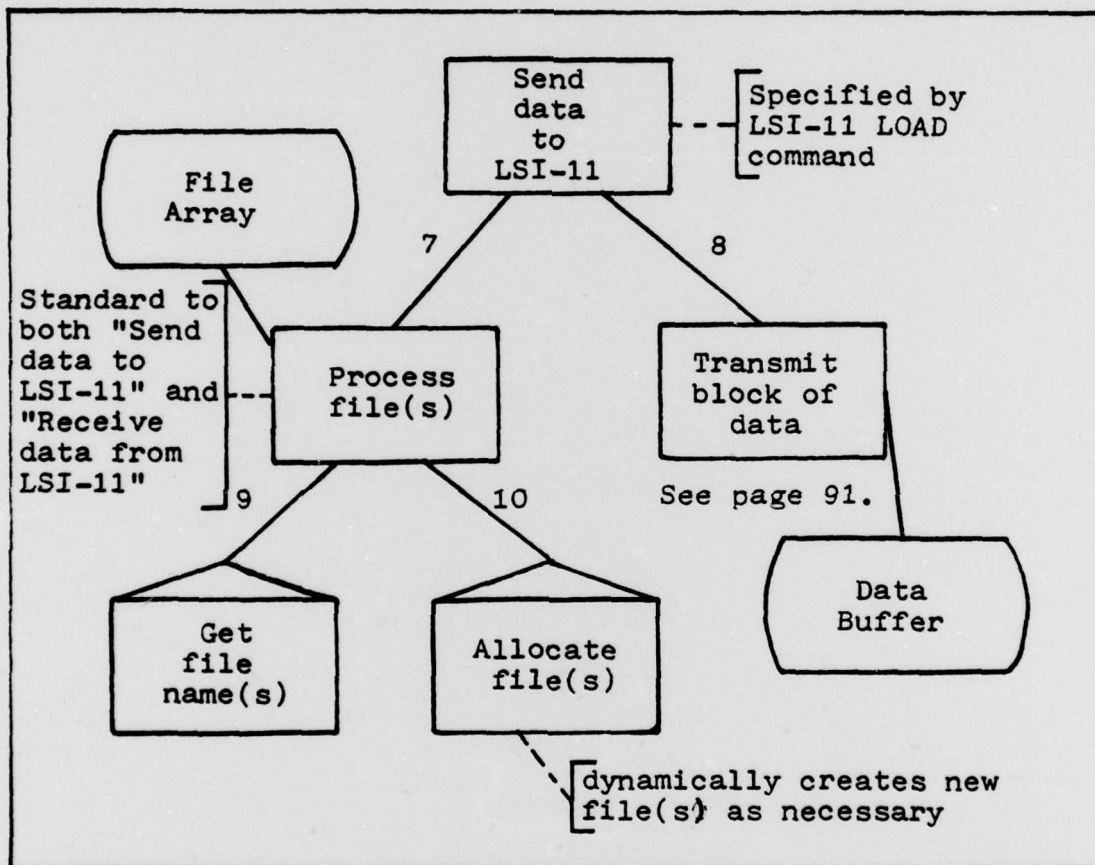


Figure 13. (cont'd)

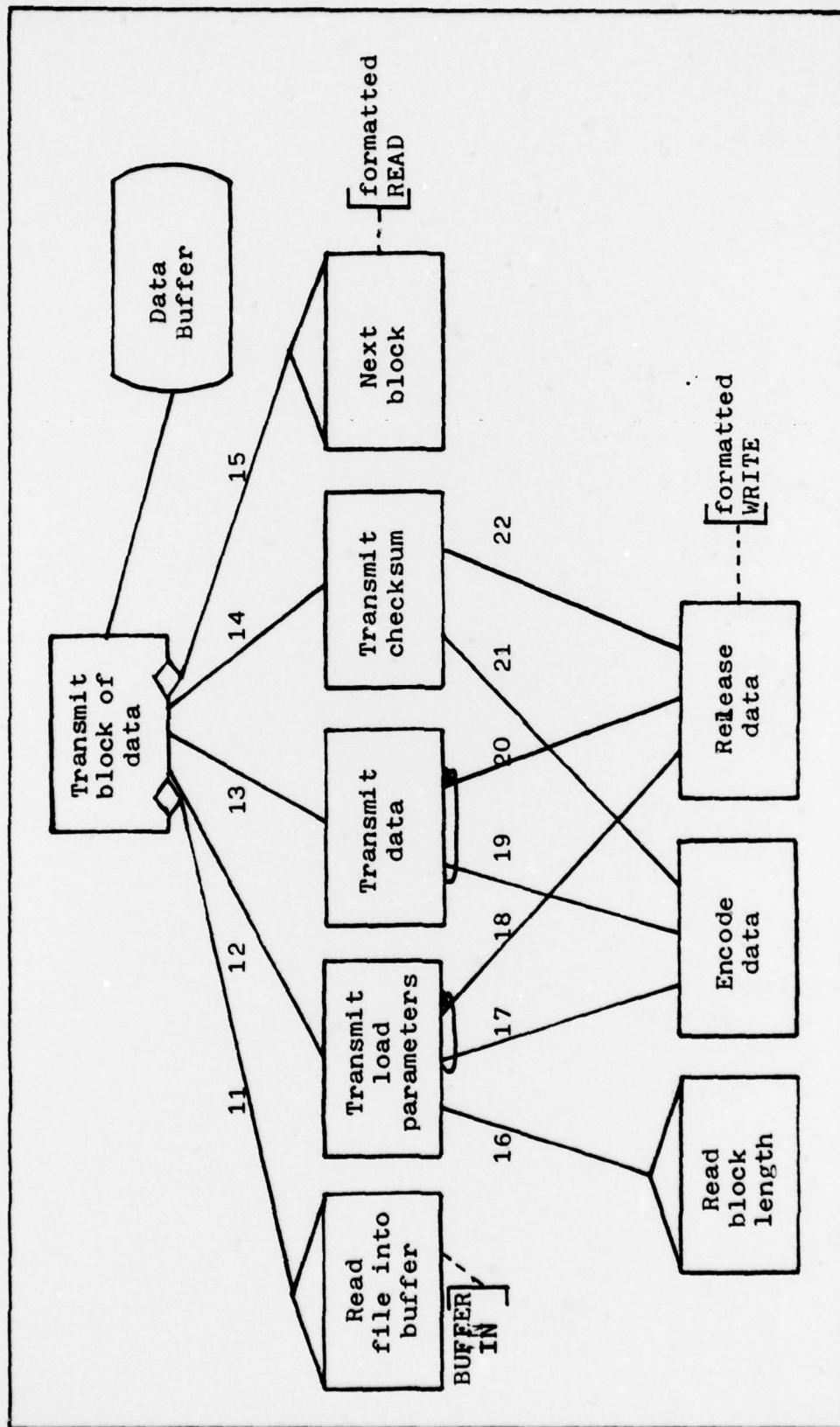


Figure 13. (cont'd)

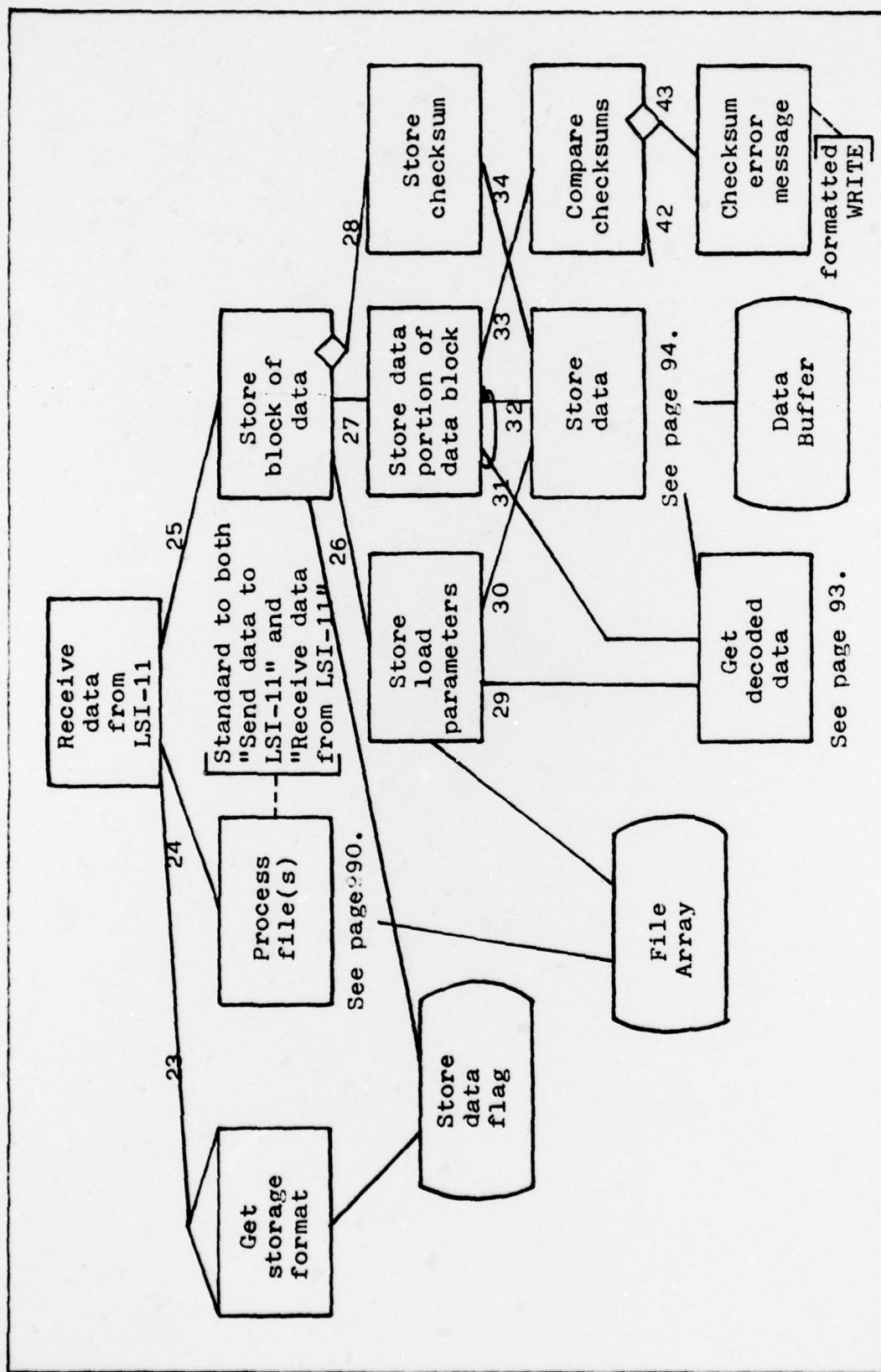


Figure 13. (cont'd)

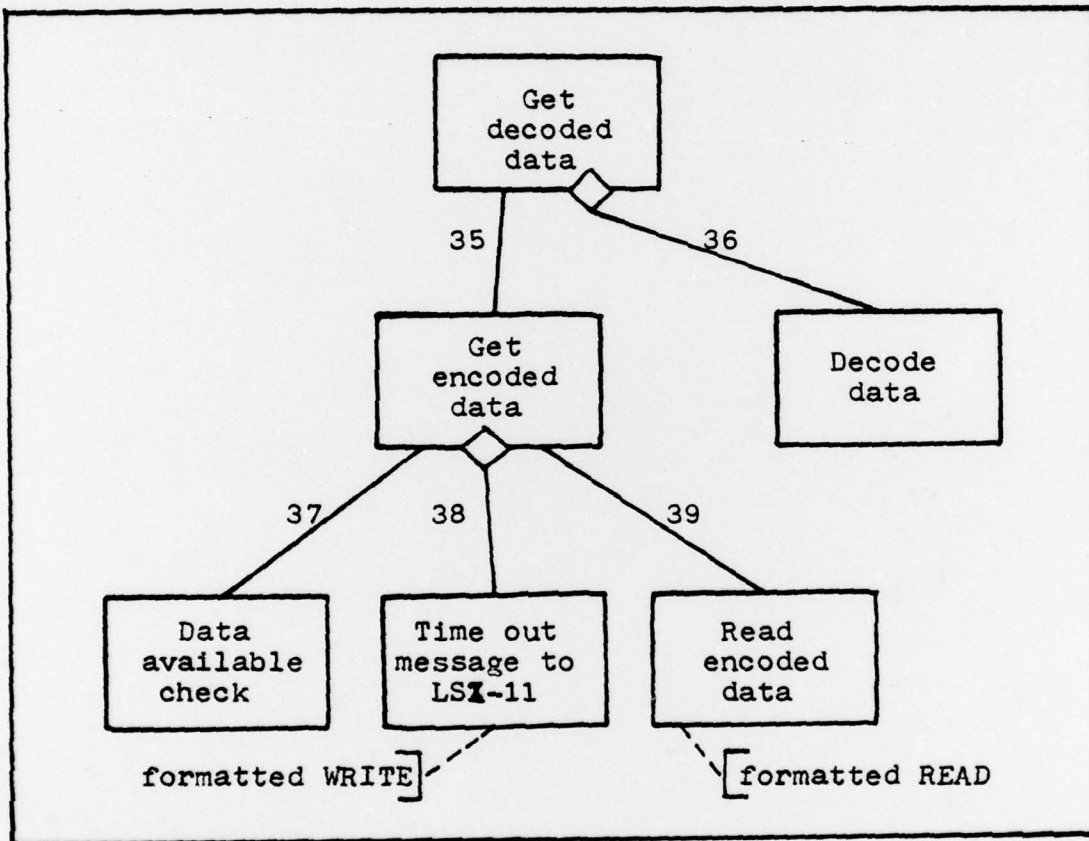


Figure 13. (cont'd)

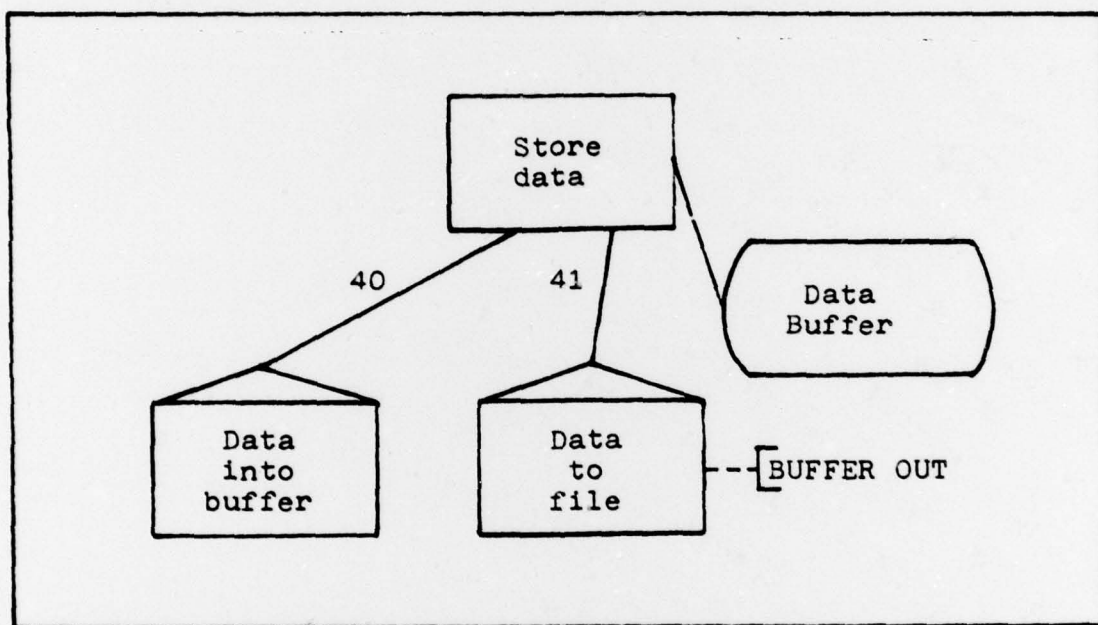


Figure 13. (cont'd)

Table VIII

SEL 86 LSINTR Data
and Control Flow

	INPUT	OUTPUT
1	-	-
2	-	-
3	-	command type
4	command type	-
5	-	-
6	-	-
7	-	-
8	-	-
9	-	file name
10	file name	-
11	data from file	-
12	-	number of bytes
13	-	-
14	-	-
15	-	<u>stop flag</u>
16	-	block length
17	Load parameter	encoded load parameter
18	encoded load parameter	-
19	data	encoded data
20	encoded data	-
21	checksum	encoded checksum
22	encoded checksum	-

Table VIII (continued).

	INPUT	OUTPUT
23	-	-
24	-	-
25	-	-
26	-	file name, number of bytes, computed checksum
27	file name, number of bytes computed checksum, block number	-
28	checksum, file name	-
29	-	decoded load parameter
30	decoded load parameter, file name	-
31	-	decoded data
32	decoded data, file name	-
33	computed checksum, block number	-
34	checksum, file name	-
35	-	encoded data
36	encoded data	decoded data
37	-	-
38	-	-
39	-	encoded data
40	decoded data	-
41	file name	-
42	-	checksum
43	block number	-

Description of SEL 86 LSINTR

SEL 86 LSINTR - controls all SEL 86 functions of the SEL 86/LSI-11 Interface Monitor. Controls initialization of needed work area and all interface operations with the LSI-11.

Initialize parameters - sets up two arrays as part of working area.

Data Buffer - only contents of this module is a buffer area to store data before it is processed.

File Array - only contents of this module is a work area array for storing names of SEL 86 files to be accessed.

Interface with LSI-11 - controls interface with LSI-11. Determines SEL 86 mode, i.e. send or receive data, and performs its functions accordingly.

Get mode - gets SEL 86 type, i.e. send or receive data from LSI-11 using formatted READ.

Dispatch on mode type - dispatches the mode type for processing as either send or receive data.

Send data to LSI-11 - controls transmission of data from a SEL 86 file.

Process file(s) - controls allocating and blocking SEL 86 files that are to be accessed. Dynamically creates new files as necessary.

Get file name(s) - gets SEL 86 file name(s) from LSI-11 using formatted READ.

Allocate file(s) - allocates and blocks SEL 86 file(s) being accessed. Dynamically creates new files as necessary.

8 Transmit block of data - controls transmission of all data on the file, a block at a time. Controls filling of storage buffer with data from file. Monitors stop flag to control transmission of each block of data.

Read file into buffer - read data from file into data buffer using BUFFER IN.

Transmit load parameters - controls release of encoded load parameters. Determines number of bytes to be transmitted.

Transmit data - controls transmission of encoded data.

Transmit checksum - controls transmission of encoded checksum.

Next block - determines if next block of data is to be transmitted by reading notification from LSI-11 in formatted READ.

Read block length - determines number of bytes to be transmitted by reading the number of bytes parameter specified in the load block.

Encode data - encodes each four bits of data into an ASC11 numeric character.

Release data - send encoded data to LSI-11 using formatted WRITE.

Receive data from LSI-11 - controls receipt of data from an LSI-11.

Get storage format - determines how data is to be stored, i.e. binary format or absolute binary load format, using a formatted READ.

Store data flag - only contents of this module is a flag specifying how the data is to be stored.

Process files - see "Process files" above.

Store block of data - controls storage of data block. Notifies LSI-11 of checksum errors specifying block number.

Store load parameters - controls determining values of load parameters and their storage on SEL 86 file if appropriate. Keeps running tally of the checksum.

Store data portion of data block - controls storage of actual data block 9 no load parameters or checksum) on SEL 86 file. Keeps a running tally of the checksum. Monitors checksum errors.

Store checksum - controls storage of checksum on SEL 86 file if appropriate.

Get decoded data - controls receiving and decoding data from LSI-11.

Decode data - decodes ASC11 characters (encoded data) into binary data.

Data available check - loop of executing statements to count length of time waiting for data from LSI-11.

Time-out message - prints warning message at LSI-11 user's console (using a formatted WRITE) that no data has been received at the SEL 86 in a specified time.

Read data - receives encoded data using formatted READ.

Store data - controls storage of data into buffer and then onto SEL 86 file.

Data into buffer - reads decoded data into Data Buffer.

Data to file - writes decoded data onto SEL 86 file
using BUFFER OUT.

Compare checksums - tests for checksum error.

Checksum error message - prints checksum error message
(specifying block number) at LSI-11 user's console using a
formatted WRITE.

Vita

Capt Janet E. Baralli was born on 4 January 1950 in Hammond, Indiana. She graduated from high school in Lansing, Illinois in 1968. In 1972 she graduated magna cum laude from Saint Mary's College in Notre Dame, Indiana where she received a Bachelor of Science degree in Chemistry. Upon graduation she entered Officer Training School at Lackland AFB and was commissioned a 2nd Lieutenant in the U. S. Air Force on 28 August 1972. Before entering AFIT, her Air Force career was spent in ADCOM as a space systems analyst. She worked in the NORAD Cheyenne Mountain Complex in addition to spending a year remote at a radar site in Turkey. She entered the School of Engineering, Air Force Institute of Technology, in June 1977.

Permanent address: 3604 Adams Street
Lansing, IL 60438


REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/78-9	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DESIGN OF A SEL 86/LSI-11 INTERFACE MONITOR		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
7. AUTHOR(s) Janet E. Baralli Capt		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, OH 45433		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Computer Activities Office (AFML/DOC)		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE December 1978
		13. NUMBER OF PAGES 103
		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Approved for public release; IAW AFR 190-17 JOSEPH P. HIPPS, Major, USAF Director of Information 1-23-79		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Software Engineering LSI-11 Structured Design Minicomputer Network Interface Monitor SEL 86		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Air Force Materials Laboratory (AFML) uses LSI-11 micro-computers as one of several computer systems available for collecting test data. For conducting these tests, LSI-11 programs must be loaded into and data collected from the LSI-11 using paper tapes. Data is later stored on a larger computer system at AFML, the SEL 86. <i>next page</i>		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

The purpose of this investigation has been to design a SEL 86/LSI-11 interface that will automate manual procedures. The interface design enhances the current LSI-11 system by providing the following capabilities: load binary programs and data residing on a SEL 86 file into LSI-11 memory; transmit data stored in LSI-11 memory to one or more SEL 86 files, and place the LSI-11 memory into a transparency mode such that it is a peripheral as viewed by the SEL 86.

The principles of software engineering have been applied in both the analysis and design phases. Formal tools have been used in defining the requirements and developing the structured design. The resulting design is an interface monitor with software residing on both the LSI-11 and the SEL 86. The added capabilities are provided using either a series of commands entered at the LSI-11 console or as call statements in LSI-11 FORTRAN compiled programs.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)